MSc thesis

Master's Programme in Computer Science

Reasoning over Assumption-Based Argumentation Frameworks via Answer Set Programming

Tuomo Lehtonen

December 10, 2019

Faculty of Science

University of Helsinki

Supervisors

Prof. Matti Järvisalo, Dr. Johannes P. Wallner

Examiners

Prof. Matti Järvisalo, Dr. Johannes P. Wallner

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: https://www.helsinki.fi/en/computer-science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Tuomo Lehtonen

Työn nimi — Arbetets titel — Title

Reasoning over Assumption-Based Argumentation Frameworks via Answer Set Programming

Ohjaajat — Handledare — Supervisors

Prof. Matti Järvisalo, Dr. Johannes P. Wallner

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
MSc thesis	December 10, 2019	50 pages

Tiivistelmä — Referat — Abstract

Formal argumentation is a vibrant research area within artificial intelligence, in particular in knowledge representation and reasoning. Computational models of argumentation are divided into abstract and structured formalisms. Since its introduction in 1995, abstract argumentation, where the structure of arguments is abstracted away, has been much studied and applied. Structured argumentation formalisms, on the other hand, contain the explicit derivation of arguments. This is motivated by the importance of the construction of arguments in the application of argumentation formalisms, but also makes structured formalisms conceptually and often computationally more complex than abstract argumentation.

The focus of this work is on assumption-based argumentation (ABA), a major structured formalism. Specifically we address the relative lack of efficient computational tools for reasoning in ABA compared to abstract argumentation. The computational efficiency of ABA reasoning systems has been markedly lower than the systems for abstract argumentation. In this thesis we introduce a declarative approach to reasoning in ABA via answer set programming (ASP), drawing inspiration from existing tools for abstract argumentation. In addition, we consider ABA$^+$, a generalization of ABA that incorporates preferences into the formalism. The complexity of reasoning in ABA$^+$ is higher than in ABA for most problems. We are able to extend our declarative approach to some ABA$^+$ reasoning problems. We show empirically that our approach vastly outperforms previous reasoning systems for ABA and ABA$^+$.

ACM Computing Classification System (CCS)
Computing methodologies → Artificial intelligence → Knowledge representation and reasoning
Theory of computation → Logic → Constraint and logic programming

Avainsanat — Nyckelord — Keywords

Argumentation, Assumption-Based Argumentation, Answer Set Programming

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Algorithms study track

Contents

1 Introduction

Formal argumentation is a vibrant research area in artificial intelligence, in particular within knowledge representation and reasoning [3, 4]. The aim is to draw conclusions from internally inconsistent or incomplete knowledge bases using formalisms in which reasoning can be done algorithmically. Thus in contrast to for example classical logic, formal argumentation represents defeasible reasoning, where new information may revoke previously reached conclusions. Various formalisms of non-monotonic reasoning have in fact been shown to be captured by an argumentative approach [22]. In addition to providing a different approach to other formalisms, formal argumentation has applications in various domains [1], such as legal, medical [16, 18], and e-government, as well as multi-agent systems [31].

Argumentation formalisms divide into abstract and structured approaches. The main model to abstract argumentation is provided by abstract argumentation frameworks (AFs), where the components are atomic arguments and attacks between arguments [22]. That is, the structure of arguments is not visible. Attacks signify conflicts between arguments: an argument that attacks another can be seen as a counterargument to the argument it attacks. The power of abstract argumentation is in its simplicity: an AF is a graph where the nodes are the arguments and the edges are the attacks. The basic reasoning mode for AFs is to determine sets of arguments that are jointly acceptable. Which sets of arguments are acceptable depend only on the attack relation and the chosen criteria for accepting arguments, the latter of which is called semantics. There are multiple central semantics, and the choice of semantics depends on the application. For example, a set of arguments is acceptable according to the admissible semantics if the set attacks every argument that attacks it, and does not attack itself.

In contrast to AFs, in structured argumentation formalisms the structure of arguments is explicit in the formalism and depends on other elements, usually a form of premises and a deductive system [7]. Attacks are also typically constructed from other elements of the formalism, such as a notion of what contradicts a specific premise. Making the structure of arguments and attacks explicit is a more realistic way of representing an argumentative scenario. Intuitively arguments most often have multiple components, such as premises, supporting facts, and rules for deriving new facts. What makes an argument conflict with another one also depends on the components of each argument. Due to the conceptual richness of structured formalisms, a knowledge base may be easier to represent using a structured argumentation formalism instead of first applying other methods to construct arguments and attacks and then representing them as an AF. In other words, structured argumentation formalisms allow for representing the knowledge base in a format more closely resembling the data. The explicitness of the structure of arguments and attacks makes the complexity of reasoning in structured argumentation formalism in many cases higher than it is in abstract argumentation [28]. There is a multitude of proposed structured argumentation formalisms, including assumption-based argumentation (ABA) [9, 20, 23, 61], ASPIC$^+$ [50,

51, 54], Defeasible Logic Programming (DeLP) [36, 35], deductive argumentation [6, 5], and Carneades [43].

We focus on ABA in this thesis. An ABA framework consists of assumptions, rules, sentences and contraries. Deductions of sentences are made from assumptions via the rules, and assumptions can be attacked by the contrary of the assumptions, contraries being sentences. Whereas in abstract argumentation sets of *arguments* are jointly accepted according to semantics, in ABA the semantics sanction acceptable sets of *assumptions*. Attacks are defined between sets of assumptions: if a set of assumptions derives the contrary of some assumption, the set of assumptions attacks this assumption and all assumption sets containing the attacked assumption. Similarly to abstract argumentation, ABA has various semantics. An acceptable set of assumptions is such that the assumptions together do not derive a contrary of any of the assumptions in the set, and in addition fulfil some further criteria enforced by the chosen semantics. Some applications of ABA include medical decision making [16, 18], decision making in a multi-agent context [31], and game theory [30].

The development of reasoning systems for argumentation formalisms has focused more on abstract argumentation than on ABA or other structured formalisms [13]. There are multiple efficient implementations of reasoning with AFs, including specialized algorithms and declarative methods. A competition for AF reasoning systems (International Competition on Computational Models of Argumentation, ICCMA) has been held every two years since 2015 [59, 34, 44]. Notably, the most efficient solvers are usually based on a declarative approach [13]. In declarative programming, a problem is modelled as another formalism, solved with a solver designed for that other formalism, and the answer to the original problem is interpreted from the solution. For example, approaches based on propositional satisfiability (SAT) [8] solving and answer set programming (ASP) [42, 52, 10, 39] have been widely used for reasoning in abstract argumentation [13]. This is motivated by a number of central reasoning problems in abstract argumentation being NP-hard [28] and by SAT and ASP being successful tools for solving NP-hard problems (NP-complete problems directly and problems beyond NP via iterative calls to solvers). These approaches consist of encoding the problem as propositional logic or as a logic program and using a SAT or an ASP solver, respectively, to solve the encoded instance. The answer to the original problem instance can be interpreted from the answer to the encoded instance. The solver technology for both of these is similar. Clingo, a state-of-the-art system, incorporates most of the techniques of modern conflict-driven clause learning SAT solvers [37].

In contrast to abstract argumentation, there are relatively few reasoning systems for ABA [13] and no solver competitions. The state-of-the-art systems for ABA are abagraph [14] and aba2af [46]. Abagraph implements a specialized algorithm for certain semantics and problems of ABA while aba2af translates ABA frameworks to AFs in order to make use of AF solvers. However, neither of the systems are purely declarative; in translation-based systems the AF part can be solved declaratively, but the translation is a non-trivial algorithm in itself. The resulting AF can also be much larger than the input ABA framework. Since many central ABA reasoning problems are NP-hard (see Section 2.2), similar declarative approaches as the ones developed for AF reasoning can conceivably be developed for ABA. The main

contribution of this work is introducing a direct and efficient declarative approach to reasoning in ABA, drawing inspiration from the success of reasoning systems for AFs. In particular, we develop ASP-based algorithms for the central ABA semantics admissible, complete, preferred, stable and ideal. This allows for the use of efficient ASP solvers to answer central NP-hard ABA reasoning problems. We show via extensive empirical experimentation that our approach is vastly more efficient than previous state-of-the-art ABA reasoning systems.

Beyond plain ABA, we also address the challenge of reasoning in a formalism extending ABA with preferences over assumptions, namely ABA$^+$ [17, 2, 19]. Preferences can be considered an important part of argumentative reasoning as they make it possible to include additional qualifying information about the situation that is being modelled, such as the plausibility of a fact or the wishes of an agent. An example of the latter is a study where (a modification of) ABA$^+$ was used to determine a set of recommendations to follow from clinical guidelines [18]. Preferences were used to represent the wishes of patients about their treatment: a patient might wish, for example, to avoid intense exercise. Then a treatment without intense exercise is chosen over ones involving it, if possible.

There are several approaches to augmenting ABA with preferences. They can be classified into meta-level and object-level approaches [20]. Object-level approaches describe a way to embed preferences into ABA frameworks by using the existing components (assumptions, rules, contraries). Thus these approaches are captured by any computational tool capable of reasoning over ABA frameworks, such as the one we introduce in this work. Meta-level approaches, on the other hand, add preferences into ABA as a new component of the formalism and alter the semantics in some way to respect the preferences. Two such formalisms are ABA$^+$, which adds preferences over assumptions and alters the attack relation, and a p_ABA [62], which adds preferences over sentences and ranks the accepted assumption sets accordingly. We consider the computation of reasoning problems of ABA$^+$ in this thesis.

The complexity of reasoning in ABA$^+$ is shown to be higher than in ABA for certain semantics [47]. However, for stable semantics, the complexity is the same. We are able to show a similar ASP encoding for reasoning over stable semantics of ABA$^+$ as for the ABA counterpart. We also introduce an ASP-based algorithm for reasoning in ABA$^+$under grounded semantics. There is one previous system implementing reasoning for ABA$^+$, called ABAplus [2]. This system supports most central semantics by translating the given ABA$^+$ framework into an AF in a way that respects the preferences. We show that the ASP approach we introduce for the two ABA$^+$ semantics significantly outperforms ABAplus.

This thesis is structured as follows. Section 2 introduces and formally defines ABA and ABA$^+$, as well as their semantics and reasoning problems. Section 3 reviews the current state-of-the-art systems for reasoning in ABA and ABA$^+$. Section 4 shows some properties of ABA and ABA$^+$ that form the basis of our encodings. Section 5 gives an overview on answer set programming. Sections 6 and 7 present our algorithms for reasoning in ABA and ABA$^+$, respectively, by encoding their reasoning problems as ASP. In Section 8, our declarative algorithms are evaluated empirically against the previous state-of-the-art systems. Finally, Section 9 concludes the thesis.

The original contributions of this work are showing properties that form the theoretical basis for our ASP encodings (Section 4), introducing ASP-based algorithms for all central ABA semantics (Section 6), introducing ASP-based algorithms for two central ABA$^+$ semantics (Section 7), and evaluating the algorithms empirically (Section 8). Some of these contributions have been published at AAAI 2019 [48] with an extended version currently at manuscript stage.

2 Assumption-based Argumentation

We give an overview of the central structured argumentation formalism assumption-based argumentation (ABA) [9, 61, 20] and its extension ABA$^+$ [17, 19, 2], which equips ABA with preferences over assumptions.

The basic elements of ABA are sentences, rules, assumptions and contraries. Rules dictate how to derive sentences from other sentences. Assumptions are sentences that are *prima facie* true. The acceptance of an assumption can be revoked by an attack to the assumption. Attacks are constructed on the basis of contraries, each assumption having one. Contraries are sentences. Intuitively, an assumption and its contrary can not hold at the same time.

There are multiple semantics for ABA, defining sets of assumptions that are jointly acceptable in a given ABA framework. The semantics are defined on the basis of attacks, and generally the semantics are requirements on the kinds of incoming and outgoing attacks of the accepted assumption set. Specifically, sufficient outgoing attacks and sufficiently restricted incoming attacks are required. Many semantics are defined in terms of other semantics, for example by further adding a subset-maximality constraint.

2.1 ABA

We recall the basic definitions for ABA, including a deductive system, an ABA framework, the notion of attack and defence, and the major semantics that we consider in this work.

A deductive system consists of a set of sentences and a set of rules. The rules specify how sentences can derive other sentences.

Definition 1 (Deductive system)**.** A *deductive system* is a pair $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a formal language, that is, a set of sentences, and $\mathcal{R}$ a set of inference rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has the form $a_0 \leftarrow a_1, \ldots, a_n$ with $a_i \in \mathcal{L}$. We denote the head of rule r by $head(r) = \{a_0\}$ and the (possibly empty) body of r with $body(r) = \{a_1, \ldots, a_n\}$.

An ABA framework contains a deductive system, a set of assumptions, and contraries to assumptions.

Definition 2 (ABA framework)**.** An *ABA framework* is a tuple $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ with $(\mathcal{L}, \mathcal{R})$ a deductive system, a set of assumptions $\mathcal{A} \subseteq \mathcal{L}$, and a contrary function $^-$ mapping assumptions $\mathcal{A}$ to sentences $\mathcal{L}$.

The contrary of an assumption $a \in \mathcal{A}$ is denoted by $\overline{a}$.

We assume that each set of an ABA framework is finite. ABA frameworks in which assumptions do not occur at the head of rules are *flat*. We restrict ourselves to flat frameworks in this study, as is usual. In particular, all state-of-the-art ABA reasoning systems are designed for

flat frameworks [46, 14]. The complexity of reasoning in general ABA frameworks is higher than in flat frameworks under many semantics [28], and thus it is not trivial to generalize algorithms developed for flat frameworks.

There are different definitions for the precise manner in which sentences are derived from a set of assumptions and a set of rules. Two notable formulations, which are equivalent for defining ABA attacks and semantics, are tree-derivability (here denoted by $\models$) and forward-derivability (here denoted by $\vdash$) [24, 25]. In both of them the derivation of a sentence is done by chaining rules so that the derived sentence is at the end the chain. Forward-derivability forms the basis of our ASP encodings (in Section 6), and we will mainly use it in our treatment of ABA. In contrast, ABA$^+$ handles preferences explicitly via tree-derived arguments, and in general, forward-derivability can not be used as an alternative notion in ABA$^+$. However, we will show in Section 2.3 that under particular semantics tree-derivability and forward-derivability are equivalent for ABA$^+$. This allows for similar ASP encodings for ABA$^+$ to those we develop for ABA under specific semantics. We will define forward-derivations now and leave tree-derivations for Section 2.3.

Definition 3 (Forward-derivation)**.** A sentence $a \in \mathcal{L}$ is *forward-derivable* from a set $A \subseteq \mathcal{A}$ via rules $\mathcal{R}$, denoted by $A \vdash_{\mathcal{R}} a$, if and only if there is a sequence of rules $(r_1, \ldots, r_n)$ where $head(r_n) = a$ and for each rule r_i, $r_i \in \mathcal{R}$, and each sentence in the body of r_i is either the head of a rule earlier in the sequence, or is in A. Formally expressed, it is required that $body(r_i) \subseteq A \cup \bigcup_{j<i} head(r_j)$ holds.

In other words, a sentence is forward-derivable from a set of assumptions via a set of rules if it can be derived by iteratively applying the rules such that every body element of a given rule is either among the given assumptions or derived earlier. The assumption set A is said to support the derivation. The deductive closure of $A \subseteq \mathcal{A}$ with respect to rules $\mathcal{R}$ is denoted by $Th_{\mathcal{R}}(A) = \{a \mid A \vdash_{\mathcal{R}} a\}$. When the set of rules available for the derivation is not relevant, we will write $\vdash$ and assume all of the rules of the framework are available (though not necessarily all needed for the derivation).

Example 1. Consider the deductive system

$$\mathcal{L} = \{a, b, c, x, y, z\},$$
$$\mathcal{R} = \{(x \leftarrow a, b), (y \leftarrow c), (z \leftarrow a, c)\}.$$

The following forward derivations, for example, are possible in the system:

$$\{a, b\} \vdash x,$$
$$\{a, c\} \vdash z,$$
$$\{a, c\} \vdash y.$$

△

ABA semantics, that is, criteria for a set of assumptions being acceptable, are defined in terms of sets of assumptions, the sentences they derive, and the assumptions that they

attack. More precisely, if a set of assumptions derives the contrary of another assumption, the set is said to attack the assumption. The notion of attack is lifted to targeting sets of assumptions by defining that the attacking set attacks all assumption sets that contain the attacked assumption. Finally, if a set of assumptions attacks all attackers of a set of assumptions, the former set defends the latter.

Definition 4 (Attack and defence)**.** Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABA framework, $A, B \subseteq \mathcal{A}$, and $c \in \mathcal{A}$.

- A *attacks* c if and only if $A \vdash \overline{c}$,
- A *attacks* B if and only if A attacks some assumption in B, and
- A *defends* B if and only if A attacks each $C \subseteq \mathcal{A}$ that attacks B.

For an assumption set being acceptable, not attacking itself is a minimum requirement.

Definition 5 (Conflict-free assumption set)**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABA framework. An assumption set $A \subseteq \mathcal{A}$ is *conflict-free* if and only if A does not attack itself.

Example 2. Let us complete the deductive system of Example 1 into an ABA framework:

$$\begin{aligned} \text{sentences } \mathcal{L} &= \{a, b, c, x, y, z\} \\ \text{assumptions } \mathcal{A} &= \{a, b, c\} \\ \text{contraries} \quad & \overline{a} = y,\ \overline{b} = z,\ \overline{c} = x \\ \text{rules } \mathcal{R} &= \{(x \leftarrow a, b), (y \leftarrow c), (z \leftarrow a, c)\} \end{aligned}$$

See Figure 2.1 for an illustration of this framework. In this framework, the set $\{a, b\}$ attacks c by deriving x, $\{c\}$ attacks a via y, and $\{a, c\}$ attacks b via z, as well as a via y. Therefore, for example $\{a, c\}$ is not conflict-free, while $\{a, b\}$ is. △

There are multiple different ABA semantics, each placing different constraints on the acceptability of assumption sets [20]. Here we define the more commonly studied ones.

Definition 6 (Semantics)**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, {}^-)$ be an ABA framework. A conflict-free set $A \subseteq \mathcal{A}$ is

- *admissible* if and only if A defends itself,
- *complete* if and only if A is admissible and contains every assumption defended by A,
- *grounded* if and only if A is the intersection of all complete assumption sets,
- *preferred* if and only if A is a subset-maximal admissible assumption set,
- *stable* if and only if each $x \in \mathcal{A} \setminus A$ is attacked by A, and

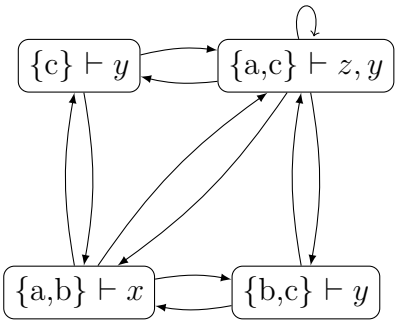

Figure 2.1: The ABA framework of Example 2 illustrated. The assumption sets of the framework as well as the sentences they can derive are shown inside the rectangles. Attacks between assumption sets are represented by the edges. For simplicity, the self-attacking assumption set $\{a, b, c\}$ is not drawn, and neither are the sets $\{a\}$ and $\{b\}$, both of which attack nothing but are attacked. The empty set, not attacking anything or being attacked, is also omitted.

- *ideal* if and only if A is the maximal admissible assumption set that is a subset of every preferred assumption set.

The term σ-assumption set is used for an assumption set under a semantics $\sigma \in \{adm, com, grd, stb, prf, ideal\}$, that is, admissible, complete, grounded, stable, preferred, and ideal semantics, respectively.

Example 3. In the ABA framework of Example 2, the different semantics accept the following assumption sets.

$$\begin{aligned} \text{admissible: } & \emptyset, \{a, b\}, \{b, c\}, \{c\} \\ \text{complete: } & \emptyset, \{a, b\}, \{b, c\} \\ \text{preferred: } & \{a, b\}, \{b, c\} \\ \text{stable: } & \{a, b\}, \{b, c\} \\ \text{ideal: } & \emptyset \\ \text{grounded: } & \emptyset \end{aligned}$$

The rationale for the σ-assumption sets are as follows. From $\{b, c\}$ and $\{c\}$ one can derive y, and from $\{a, c\}$ both z and y. The empty set is trivially admissible, and since the empty set alone defends nothing, it is also complete. Set $\{c\}$ is complete because it is only attacked by $\{a, b\}$ which it defends itself against by attacking a. The admissibility of $\{a, b\}$ and $\{b, c\}$ is evident: they attack everything that attacks them. Moreover, one cannot add the remaining assumption (c and a, respectively) to the sets without making them self-conflicting, and thus they are complete and preferred. They are also stable, since they attack the remaining assumption. The ideal assumption set is empty, because the subsets of every preferred

assumption sets are $\emptyset$ and $\{b\}$, of which $\{b\}$ is not admissible. The grounded assumption set is also empty, as the intersection of all complete assumption sets is empty. $\triangle$

Flat ABA frameworks satisfy a property called Fundamental Lemma, stating that given an admissible assumption set A and an assumption b that is defended by A, $A \cup \{b\}$ is admissible [9]. A consequence of this is that the grounded assumption set can be computed as the least fixed point of a defence operator. Computing the grounded assumption set this way is more direct than computing all complete assumption sets and taking their intersection. The defence operator is defined as $def_F(A) = \{a \in \mathcal{A} \mid A \text{ defends } a\}$.

Proposition 1 ([9, Theorem 6.2])**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. The least fixed point of def_F is the grounded assumption set of F.

2.2 ABA Reasoning Problems

A basic reasoning task of ABA is the enumeration all assumption sets accepted under a given semantics.

Definition 7 (σ-assumption set enumeration)**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework and σ a semantics. Report all σ-assumption sets.

Instead of reporting all accepted assumption sets under a semantics, often a relevant question is to find out whether a given sentence is acceptable under a semantics. To answer this question, there are two prominent reasoning modes. One can either accept sentences derivable in any σ-assumption set, or only sentences that are derivable in all σ-assumption sets.

Definition 8 (Acceptability of sentences)**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework and a semantics σ. A sentence $s \in \mathcal{L}$ is

- *credulously accepted* under semantics σ if and only if there is a σ-assumption set A so that $A \vdash s$, and
- *sceptically accepted* under semantics σ if and only if $A \vdash s$ for all σ-assumption sets A.

Example 4. One can check the acceptability status of sentences of our running example from the σ-assumption sets listed in Example 3. All of the assumptions are credulously accepted under admissible, complete, preferred, and stable semantics. On the other hand, nothing is credulously or sceptically accepted under ideal or grounded semantics. Under preferred and stable semantics, b is sceptically accepted. Nothing is sceptically accepted under admissible and complete semantics, as $\emptyset$ is admissible and complete. $\triangle$

Credulous reasoning coincides under admissible, complete, and preferred semantics [9, 20]. Intuitively, this follows from the facts that any admissible assumption set is a subset of some complete and some preferred assumption set, any complete assumption set is admissible and a subset of some preferred assumption set, and finally any preferred assumption set is admissible

and complete. For ideal and grounded semantics, credulous and sceptical reasoning both amount to finding the unique σ-assumption set of the given semantics (and checking if the queried assumption is contained in it).

The computational complexity of credulous and sceptical acceptance under different semantics is well-established. These are decision problems: their answer is either YES or NO. We will recall the characterisations of the complexity classes relevant here before discussing the complexity of reasoning in ABA. For more details on computational complexity, see [53].

The class P contains problems that can be solved in polynomial time with a deterministic Turing machine. The class NP contains problems that can be solved in polynomial time with a nondeterministic Turing machine. Equivalently, for any problem in NP, a (polynomial-sized) witness of a YES-instance can be verified in polynomial time. Witness here means essentially a guessed solution; in the context of credulous reasoning in ABA, a witness is a set of assumptions. The class coNP contains problems whose complement is in NP.

For a given complexity class C, a C-hard problem is at least as hard as any problem in C. More precisely, if every problem in C can be reduced in polynomial time to a problem P, then P is C-hard. A problem is said to be C-complete if it is C-hard and also contained in C.

A C-oracle is a procedure that solves a C-complete problem in constant time. With this, complexity classes higher in the polynomial hierarchy can be defined. The classes NP and coNP are on the first level of the hierarchy. On the second level are problems that belong to the first level when assuming access to an NP-oracle. Relevant to us are the second-level classes Σ_2^{P}, Π_2^{P}, and Θ_2^{P}. A problem is contained in Σ_2^{P} (also written as $\mathrm{NP}^{\mathrm{NP}}$) if it can be solved in polynomial time with a non-deterministic Turing machine that has access to an NP-oracle. Π_2^{P} (also written as $\mathrm{coNP}^{\mathrm{NP}}$), is the class of problems whose complements can be solved in polynomial time with a non-deterministic Turing machine that has access to an NP-oracle. Θ_2^{P} contains the problems that can be computed in polynomial time by an algorithm making a logarithmic number of calls to an NP-oracle.

The complexity of the core ABA reasoning problems is summarised in Table 2.1. The results for admissible, stable and preferred were shown in [21], for ideal in [26], and for complete and grounded in [48]. For more complexity results for ABA, see [28]. As enumerating σ-assumption sets is a function problem rather than decision problem, its complexity is not as easily analysed [28].* However, the complexity of credulous and sceptical acceptance give a good indication of the complexity of enumerating σ-assumption sets. For the semantics with a unique σ-assumption set (grounded and ideal), credulous and sceptical reasoning and enumerating σ-assumption sets coincide. For the others, enumerating σ-assumption sets is at least as hard as the harder of the two decision problems.

*There is recent research into the complexity of enumerating σ-assumption sets in a special case of ABA, called bipolar ABA [45]. Another recent study investigated the complexity of enumerating AF extensions (an extension being a set of arguments accepted under a semantics) [32].

Table 2.1: The complexity of deciding the acceptability of a sentence in ABA. For a complexity class C, C-c is an abbreviation for C-complete.

semantics	credulous	sceptical
admissible	NP-c	in P
complete	NP-c	in P
preferred	NP-c	Π_2^{P}-c
stable	NP-c	coNP-c
grounded	in P	in P
ideal	Θ_2^{P}-c	Θ_2^{P}-c

2.3 Relating Forward-derivability and Tree-derivability

The ABA$^+$ formalism is defined in terms of tree-derivability, and in general one can not substitute it with forward-derivability in ABA$^+$. In this subsection we discuss tree-derivability [23]. We recall a result on the correspondence between forward-derivability and tree-derivability: a sentence is forward-derivable from a set of assumptions if and only if the sentence is tree-derivable from a subset of the set of assumptions in question.

Unlike forward derivability, tree-derivability does not allow redundant assumptions in the underlying assumption set. The derivation sequence is viewed as a tree, so that each given assumption must be connected via the given rules to the sentence that is finally being derived. In the tree, the derived sentence is the root; the leaves are either $\top$ (signifying that the sentence is derivable from the empty set) or the assumptions from which the sentence is derived; the inner nodes are heads of rules used in the derivation; and the edges correspond to rules. A derivation tree is often called an argument.

Definition 9 (Tree-derivation)**.** A sentence $s \in \mathcal{L}$ is *tree-derivable* from a set of assumptions $A \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $A \models_R s$, if and only if there is a finite tree T such that

1. the root is labelled s,
2. the set of labels of the leaves is either A or $A \cup \{\top\}$,
3. internal nodes are labelled by some $a \in \mathcal{L}$, and as its children are the elements of the body of some rule in R such that a is the head of the rule, and
4. each rule of R is used in the tree (as described in Item 3).

When the set of rules R used in the derivation is not relevant, we will write $\models$ and leave R implicit.

Example 5. In the deductive system of Example 1, the first two forward-derivations shown in the example also serve as tree-derivations: $\{a, b\} \models x$ and $\{a, c\} \models z$. However, while $\{a, c\} \vdash y$, there is no corresponding tree-derivation, because there is no way to make use of a in deriving y from c. Instead, $\{c\} \models y$ is a tree-derivation. $\triangle$

There is a close correspondence between tree-derivability and forward-derivability.

Proposition 2 ([24, Theorem 4.1])**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework. It holds that

- if $A \models_R s$ then $A \vdash_{\mathcal{R}} s$, and
- if $A \vdash_{\mathcal{R}} s$ there is an $A' \subseteq A$ and $R \subseteq \mathcal{R}$ so that $A' \models_R s$.

In words, if there is a tree-derivation for a sentence, there is a forward-derivation for the sentence from the same set of assumptions. Moreover, if there is a forward-derivation for some sentence, there is also a tree-derivation for the sentence from some subset of assumptions and rules of the forward-derivation. To realize the latter, note that simply disregarding the assumptions and rules not actually needed in the derivation yields a tree-derivation. Thus tree-derivation is stricter and more minimal. Due to the correspondence between the derivation types, we may refer to just derivations if the type of derivation is not relevant. As noted before, the semantics of ABA can be defined equivalently via tree-derivations. It suffices to modify the definition of attack so that an assumption set A attacks an assumption b if any subset of A tree-derives $\overline{b}$ from any subset of the rules of the framework.

2.4 ABA$^+$: ABA with Preferences

Preferences can be used to resolve conflicts in favour of a more preferred conclusion. The interpretation of the preference ordering depends on the application; one example is the wishes of a patient concerning their treatment [18]. We specifically focus on ABA$^+$, which extends ABA by including preferences over assumptions. Intuitively, this helps resolve conflicts by preventing less preferred assumption sets from attacking more preferred ones by reversing attacks.

Definition 10 (ABA$^+$ framework)**.** An *ABA$^+$ framework* is a tuple $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ with $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ an ABA framework and a preorder $\leq$ over $\mathcal{A}$.

A preorder is a binary relation that is transitive and reflexive*. It is useful to distinguish the case when an assumption is strictly less preferred than another one. For this, we write $a < b$ if and only if $a \leq b$ and $b \not\leq a$, for $a, b \in \mathcal{A}$. We focus only on flat ABA$^+$ frameworks.

In ABA$^+$, the preferences modify the attack relation. As we will see, the semantics and reasoning tasks remain analogous to ABA. The attack relation after preferences are taken into account is named $<$-attacks.

Definition 11 ($<$-attack)**.** Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework. Assumption set $A \subseteq \mathcal{A}$ *$<$-attacks* $B \subseteq \mathcal{A}$ if and only if

*A binary relation R is transitive iff $R(a, b)$ and $R(b, c)$ imply $R(a, c)$ for all a, b, c in the domain of R, and it is reflexive iff $R(a, a)$ holds for all a in the domain of R.

- there is a tree-derivation $A' \models \overline{b}$, for some $A' \subseteq A$ and $b \in B$, so that $\nexists a' \in A'$ with $a' < b$, or
- there is a tree-derivation $B' \models \overline{a}$, for some $a \in A$ and $B' \subseteq B$ so that $\exists b' \in B'$ with $b' < a$.

Notice that, as noted before, attacks in ABA$^+$ are defined in terms of tree-derivability and can not in general be equivalently defined via forward-derivability. The first case in the definition yields a *normal* $<$-attack. There is a normal $<$-attack from A to B if a subset of A deriving a contrary of some assumption in B contains only assumptions that are not less preferred than the assumption in B. Attacks according to the second case are called *reverse* $<$-attacks. As the name suggests, an attack can be reversed when preferences are taken into account. Namely, if a subset B' of B tree-derives a contrary of some assumption a in A, but some assumption in B' is strictly less preferred than a, then A reverse $<$-attacks B.

Example 6. Let us add a preference relation to the ABA framework of Example 2.

$$\begin{aligned} \text{sentences } \mathcal{L} &= \{a, b, c, x, y, z\} \\ \text{assumptions } \mathcal{A} &= \{a, b, c\} \\ \text{contraries} \quad & \overline{a} = y,\ \overline{b} = z,\ \overline{c} = x \\ \text{rules } \mathcal{R} &= \{(x \leftarrow a, b), (y \leftarrow c), (z \leftarrow a, c)\} \\ \text{preferences} \quad & a < c \end{aligned}$$

Now the attacks from $\{a, b\}$ to $\{c\}$ and $\{a, c\}$ are reversed (in this case collapsed into the attack already existing in the other direction). This is due to the fact that an assumption supporting the attack, namely a, is strictly less preferred than the assumption c being attacked. △

ABA$^+$ semantics are direct generalisations of ABA semantics. First note that for determining conflict-freeness, one does not need to account for preferences: a set of assumptions is conflict-free in an ABA$^+$ framework if and only if the same set is conflict-free in the corresponding ABA framework with the preference relation emptied [17, Theorem 3.5]. This is because if a non-preference attack is present between sets $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{A}$, then either there is a normal $<$-attack in the same direction, or a reverse $<$-attack in the reverse direction. That is, a non-preference attack always translates to either kind of $<$-attack. The notion of defence in ABA$^+$ is also analogous to the one in ABA: a set $A \subseteq \mathcal{A}$ $<$-*defends* assumption set $B \subseteq \mathcal{A}$ if for all $C \subseteq \mathcal{A}$ that $<$-attack B it holds that A $<$-attacks C.

Definition 12 (ABA$^+$ semantics)**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework. A conflict-free assumption set A is

- $<$-*admissible* if and only if A defends itself,
- $<$-*complete* if and only if A is admissible and contains all assumption sets A defends,

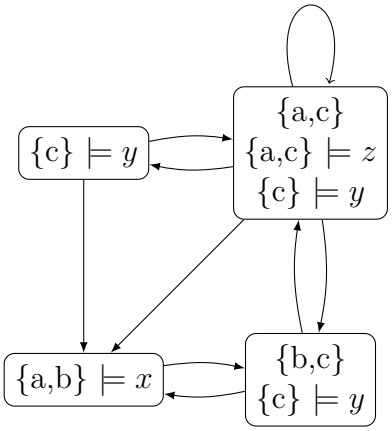

Figure 2.2: The ABA$^+$ framework of Example 6 after preferences have modified the attack relation. When needed, the assumption set is given at the top of a box and the tree-derivations from this set after it. Again for simplicity, the self-attacking assumption set $\{a, b, c\}$ is not drawn, and neither are the sets $\{a\}$ nor $\{b\}$, both of which attack nothing but are attacked.

- $<$-*grounded* if and only if A is the intersection of all $<$-complete assumption sets,
- $<$-*preferred* if and only if A is a subset-maximal $<$-admissible assumption set,
- $<$-*stable* if and only if each $\{x\} \subseteq \mathcal{A} \setminus A$ is $<$-attacked by A, and
- $<$-*ideal* is A is the maximal admissible assumption set that is a subset of every $<$-preferred assumption set.

The reasoning tasks (credulous and sceptical reasoning) for ABA$^+$ are the same as for ABA, after replacing σ with $<$-σ.

Example 7. In the ABA$^+$ framework of Example 6, the $<$-σ-assumption sets are as follows.

$$\begin{aligned} <\text{-admissible: } & \emptyset, \{b, c\}, \{c\} \\ <\text{-complete: } & \{b, c\} \\ <\text{-preferred: } & \{b, c\} \\ <\text{-stable: } & \{b, c\} \\ <\text{-ideal: } & \{b, c\} \\ <\text{-grounded: } & \{b, c\} \end{aligned}$$

The preferences settled the dilemma of choosing between mutually conflicting sets $\{a, b\}$ and $\{b, c\}$ under many semantics. Since c is more preferred than a, the set $\{b, c\}$ is solely credulously accepted under most semantics.

The rationale for the $<$-σ-assumption sets are as follows. One can derive y from both $\{c\}$ and $\{b, c\}$. The empty set is trivially $<$-admissible. Further, $\{c\}$ is $<$-admissible since it defends

itself against attacks, namely from $\{a, c\}$; and $\{c\}$ is not $<$-complete because it defends b. Instead, $\{b, c\}$ is plainly $<$-admissible and moreover the only $<$-complete and $<$-preferred assumption set because one cannot add the last assumption, a, to the set without making it self-conflicting. $\{b, c\}$ is also $<$-stable because it attacks a. Finally, $\{b, c\}$ is $<$-ideal as the $<$-admissible intersection of preferred assumption sets, and $<$-grounded as the intersection of the $<$-complete assumption sets. △

Not all properties of ABA semantics carry over to ABA$^+$ [17]. Notably, the Fundamental Lemma does not in general hold for ABA$^+$ frameworks, not even when restricted to flat frameworks. Recall that the Fundamental Lemma states that if a set of assumptions A is admissible and defends an assumption b, then $A \cup \{b\}$ is also admissible. Recall further that a consequence of the Fundamental Lemma not holding is that the grounded assumption set can not be found by computing the least fixed point of the def_F operation. However, for ABA$^+$ frameworks satisfying a property called Axiom of Weak Contraposition, the Fundamental Lemma does hold, and the $<$-grounded assumption set can be computed similarly to ABA grounded assumption set, as shown in [17].

Definition 13 (Axiom of Weak Contraposition)**.** An ABA$^+$ $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ satisfies the Axiom of Weak Contraposition (WCP) if for each $A \subseteq \mathcal{A}$ and $b \in \mathcal{A}$ it holds that

$$\begin{aligned}&\text{if } A \models \overline{b} \text{ and } \exists a' \in A \text{ so that } a' < b,\\ &\text{then } A' \models \overline{a} \text{ for some } \leq\text{-minimal } a \in A \text{ and } A' \subseteq (A \setminus \{a\}) \cup \{b\}.\end{aligned}$$

In words, WCP insists that when a set of assumptions A derives the contrary of an assumption b that is more preferred than some assumption in A, then b together with some subset of A must have a way to attack some $\leq$-minimal assumption of A. This ensures that more preferred assumptions have a way to attack less preferred ones even before attack reversals.

Example 8. The framework of Example 6 satisfies WCP. The two attacks to a more preferred assumption are from $\{a, b\}$ to c and $\{a, c\}$ via z. One can indeed tree-derive y, the contrary of the less preferred assumption, a, from a subset of $(\{a, b\} \setminus \{a\}) \cup \{c\} = \{b, c\}$ (the subset being just c), and thus the requirement is satisfied. △

The computational complexity of ABA$^+$ has been studied in [48]. Notably, it was shown that under $<$-stable semantics, credulous reasoning is NP-complete and sceptical reasoning is coNP-complete. This means that for stable semantics, adding preferences does not increase the complexity of deciding the acceptability of sentences. However, it was shown that verifying whether a given set of assumptions is $<$-admissible is coNP-complete, a jump in complexity compared to verifying admissibility in ABA, which is in P. Based on this it is shown that credulous reasoning under $<$-admissible semantics is Σ_2^{P}-hard, a higher complexity than the NP-complete credulous reasoning under admissible semantics.

This latter result is refined to show completeness in addition to hardness in [47]. We show in this thesis (in Section 4) that finding the $<$-grounded assumption set in a framework that satisfies WCP can be done with a polynomial-time algorithm with access to an NP-oracle.

3 Systems for Reasoning in ABA and ABA$^+$

This section gives an overview of the reasoning systems previously developed for ABA and ABA$^+$. In particular, we discuss the state-of-the-art systems abagraph, aba2af and ABAplus, against which we evaluate our ASP-based approach in Section 8. A summary of which reasoning problems the considered systems support is shown in Table 3.1 for ABA and in Table 3.2 for ABA$^+$. For a more thorough survey of argumentation systems, see [13].

ABA reasoning systems can be classified into three categories: translation-based systems, specialized algorithms, and direct declarative approaches. The most notable specialized systems implement algorithms for so-called dispute derivations, which are procedures for determining the credulous acceptability of a given sentence under certain semantics [60, 15, 14]. The most recent dispute derivation system is abagraph [14], improving on the earlier systems grapharg [15], proxdd [60] and CaSAPI [33], all implemented in Prolog. Abagraph can reason credulously under admissible and grounded semantics, and also enumerate solutions.

Another specialized approach is implemented in TweetyProject, a Java library of various approaches to logical aspects of artificial intelligence and knowledge representation [58, 57]. Tweety implements σ-assumption set enumeration under admissible, complete, stable, preferred, ideal and grounded semantics.

The translation-based approaches translate the input ABA frameworks to abstract argumentation frameworks and allow for the use of fast AF reasoning systems, mainly ones based on SAT or ASP solvers. A problem of translation-based systems is that the translation between ABA and AF takes time and the resulting AF instance may be much larger than the input ABA framework. For regular ABA, the system aba2af translates ABA frameworks to AFs and uses ASP encodings on the AF-level [46]. Aba2af supports credulous and sceptical reasoning under admissible, stable and preferred semantics. Abagraph and aba2af can both reason credulously under complete and preferred semantics via admissible semantics, as these tasks coincide.

A direct declarative approach encodes an ABA framework as another formalism (such as SAT or ASP) and uses a solver for that formalism to solve the problem. There is a lack of direct declarative approaches to ABA reasoning. Our work rectifies the situation by introducing ASP-based algorithms for admissible, complete, stable, preferred, and ideal semantics, allowing for credulous and sceptical reasoning as well as σ-assumption set enumeration (see Section 6). We note that recently a single logic programming translation for ABA was put forth, such that reasoning under different ABA semantics is accomplished by using different logic programming semantics [12]. The translation consists of representing the input framework in a new way. Namely, any assumption in the rule body is replaced with its contrary. The answer set semantics (called 2-valued stable semantics in [12]) corresponds to ABA stable semantics given the translation. In other words, one obtains the stable assumption sets of a given ABA framework by representing the framework in the specified way and solving the answer sets of this representation with an ASP solver. Thus restricting to stable seman-

Table 3.1: The ABA reasoning problems that the different reasoning systems support. The problems are credulous/sceptical reasoning and σ-assumption set enumeration under a given semantics. **X** marks a problem that the given system supports directly, * a problem supported via admissible semantics, and ** a problem supported via σ-assumption set enumeration.

System	**Problem**	*adm*	*com*	*stb*	*prf*	*ideal*
abagraph	credulous	**X**	*	-	*	-
	sceptical	-	-	-	-	-
	σ-a.s. enumeration	-	-	-	-	
aba2af	credulous	**X**	*	**X**	**X**	-
	sceptical	**X**	-	**X**	**X**	-
	σ-a.s. enumeration	-	-	-	-	-
ASP	credulous	**X**	**X**	**X**	**X**	**X**
	sceptical	**X**	**X**	**X**	**X**	**X**
	σ-a.s. enumeration	**X**	**X**	**X**	**X**	**X**
translation in [12]	credulous	-	-	**X**	-	-
	sceptical	-	-	**X**	-	-
	σ-a.s. enumeration	-	-	**X**	-	-
ABAplus	credulous	**	**	**	**	**
	sceptical	**	**	**	**	**
	σ-a.s. enumeration	**X**	**X**	**X**	**X**	**X**

tics of ABA, the encoding presented in [12] can be compared to the one introduced in this thesis using an ASP solver for both. Our stable encoding is stylistically uniform with our encodings for the other semantics, while the one presented in [12] can be considered simpler. The encoding of [12] has not been implemented before to the best of our knowledge.

For ABA, at present the state-of-the-art systems are abagraph [14] and aba2af [46]. We compare the performance of the encodings introduced in this work to them, as well as the stable encoding of [12]. Due to some missing semantics from what abagraph and aba2af support, we will also take ABAplus (see next paragraph) into the comparison for some ABA semantics.

The only system currently supporting ABA$^+$ reasoning is ABAplus [2]. ABAplus is a translation-based system, translating from ABA$^+$ to AF while accounting for preferences, and using Aspartix ASP encodings [29] on the AFs. ABAplus only supports σ-assumption set enumeration instead of credulous or sceptical reasoning. Credulous and sceptical queries can be answered by enumerating all σ-assumption sets, of course, but this most likely results in worse computational performance than a more direct approach. ABAplus can be used to reason over regular ABA frameworks too, because ABA$^+$ is a generalisation of ABA. ABAplus supports σ-assumption set enumeration under $<$-stable, $<$-grounded, $<$-complete, $<$-preferred and $<$-ideal semantics. ABAplus requires that WCP (recall Definition 13) is satisfied by the ABA$^+$ framework. It can modify a framework to satisfy WCP. ABAplus also places additional restrictions on the input framework: assumptions cannot be contraries and each assumption must have a unique contrary.

Table 3.2: The ABA$^+$ reasoning problems that the different reasoning systems support. For <-grounded, both systems require that WCP is satisfied in the ABA$^+$ framework. The problems are credulous/sceptical reasoning and σ-assumption set enumeration under a given semantics. **x** marks a problem that the given system supports directly and ** a problem supported via σ-assumption set enumeration.

System	**Task**	*<-com*	*<-stb*	*<-prf*	*<-grd* (WCP)	*<-ideal*
ASP	credulous	-	**X**	-	**X**	-
	sceptical	-	**X**	-	**X**	-
	σ-a.s. enumeration	-	**X**	-	**X**	-
ABAplus	credulous	**	**	**	**	**
	sceptical	**	**	**	**	**
	σ-a.s. enumeration	**X**	**X**	**X**	**X**	**X**

We provide ASP-based algorithms for the ABA$^+$ semantics <-stable and <-grounded in this thesis. Our algorithm for <-grounded requires that WCP is satisfied in the given ABA$^+$ framework.

4 Properties of ABA and ABA$^+$

We show properties of ABA and ABA$^+$ that will form the formal basis of our ASP encodings presented in Sections 6 and 7. To this end, we restate some semantics to better match the language of ASP. In addition we show complexity results for ABA$^+$, which suggest how one can, in spite of the complexity jump in many semantics of ABA$^+$, implement ABA$^+$ reasoning in ASP for $<$-grounded and $<$-stable semantics.

First, we explicate that a set of assumptions defends another set of assumptions if and only if the former set defends all assumptions in the latter set.

Lemma 1. Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework, and $A, B \subseteq \mathcal{A}$. A defends B if and only if A defends each $\{b\} \subseteq B$.

Proof. Assume that $A \subseteq \mathcal{A}$ defends $B \subseteq \mathcal{A}$. Let $C \subseteq \mathcal{A}$ attack some $b \in B$. Thus by definition C attacks B. Since A defends B, A must attack C, and thus A defends $\{b\}$. Since b is arbitrary, A defends each $\{b\} \subseteq B$.

Assume that A defends each $\{b\} \subseteq B$. Let $C \subseteq \mathcal{A}$ be such that C attacks B. Then, by definition of attacks in ABA, there is a $b' \in B$ such that $C \vdash_{\mathcal{R}} \overline{b'}$. This implies that C attacks $\{b\}' \subseteq B$. Since A defends each assumption in B, A attacks C. Therefore A defends B. □

With the help of Lemma 1 we can restate the defence of assumption set as well as admissible and complete semantics in terms of the set of assumptions attacked by a given assumption set. This helps us formulate our ASP encodings for admissible, complete and preferred semantics in Section 6.

Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework and $E \subseteq \mathcal{A}$ be a set of assumptions. Define $D_E = \{a \in \mathcal{A} \mid E \text{ attacks } \{a\}\}$. Recall that in ABA an assumption set E attacks $\{a\}$ if and only if $E \vdash_{\mathcal{R}} \overline{a}$.

Proposition 3. Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ be an ABA framework and $E \subseteq \mathcal{A}$ be a conflict-free set of assumptions. Then

1. E defends an assumption $a \in \mathcal{A}$ if and only if $\mathcal{A} \setminus D_E$ does not attack a,
2. E is admissible if and only if $\mathcal{A} \setminus D_E$ does not attack E, and
3. E is complete if and only if E is admissible and for each $b \in \mathcal{A} \setminus E$, b is attacked by $\mathcal{A} \setminus D_E$.

Proof. Item 1: By definition E defends an assumption $a \in \mathcal{A}$ if and only if A attacks C for all $C \subseteq \mathcal{A}$ that attack a. By construction of D_E, E attacks any set $X \subseteq \mathcal{A}$ for which $X \cap D_E \neq \emptyset$. Thus E defends a if and only if $\mathcal{A} \setminus D_E$ does not attack a.

Item 2: By definition E is admissible if and only if it defends itself. By the previous item and Lemma 1, E defends itself if and only if $\mathcal{A} \setminus D_E$ does not attack E.

Item 3: By definition E is complete if and only if it is admissible and every assumption that is outside of it is not defended by it. By the first item and Lemma 1, this is the case exactly when each $b \in \mathcal{A} \setminus E$ is attacked by $\mathcal{A} \setminus D_E$. □

Extending our treatment to ABA$^+$, we show a different characterisation of $<$-stable semantics that uses forward-derivability instead of tree-derivability, and later provide an algorithm for finding the $<$-grounded assumption set in a framework that satisfies WCP. These help us devising ASP-based algorithms for reasoning under these semantics.

First we show that normal $<$-attacks can be defined via forward-derivability, as well as reverse $<$-attacks from a conflict-free assumption set to a singleton assumption set.

Lemma 2. Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework, $A, B \subseteq \mathcal{A}$ be two sets of assumptions, and $b \in \mathcal{A}$.

1. Set A normally $<$-attacks B if and only if $A' \vdash_{\mathcal{R}} \overline{b}$, for some $A' \subseteq A$, $b \in B$, and $\nexists a' \in A'$ with $a' < b$.
2. If A is conflict-free, then A reversely $<$-attacks $\{b\}$ if and only if $\{b\} \vdash_{\mathcal{R}} \overline{a}$ for some $a \in A$ and $b < a$.

Proof. Item 1: by Proposition 2 it holds that if $X \models_R s$, then $X \vdash_{\mathcal{R}} s$, and if $X \vdash_{\mathcal{R}} s$, then there is an $X' \subseteq X$ and $R \subseteq \mathcal{R}$ such that $X' \models_R s$. Assume assumption set A normally $<$-attacks B. By definition of $<$-attack, there is a $A' \models_R \overline{b}$, for some $A' \subseteq A$ such that $b \in B$ and $\nexists a' \in A'$ with $a' < b$. This implies that $A' \vdash_{\mathcal{R}} \overline{b}$.

For the other direction, assume that $A' \vdash_{\mathcal{R}} \overline{b}$ for some $A' \subseteq A$, $b \in B$, and $\nexists a' \in A'$ with $a' < b$. Again by Proposition 2 there is an $A'' \subseteq A'$ such that $A'' \models_R \overline{b}$ for some $R \subseteq \mathcal{R}$. Since $\nexists a' \in A'$ such that $a' < b$ and $A'' \subseteq A'$, it follows that $\nexists a'' \in A''$ such that $a'' < b$. Thus item 1 holds.

Item 2: Assume A is conflict-free. Then $\emptyset \not\models_R \overline{a}$ for all $a \in A$. Assume that A reversely $<$-attacks $\{b\}$ and thus a subset of $\{b\}$ containing an assumption less preferred than some $a \in A$ attacks a. The only possibility is that $\{b\} \models_R \overline{a}$ for some $a \in A$ such that $b < a$. It follows by Proposition 2 that $\{b\} \vdash_{\mathcal{R}} \overline{a}$.

For the other direction, assume that A is conflict-free and $\{b\} \vdash_{\mathcal{R}} \overline{a}$ for some a such that $b < a$. By Proposition 2, there is a $B \subseteq \{b\}$ such that $B \models_R \overline{a}$. Since the only subsets of $\{b\}$ are $\{b\}$ and $\emptyset$, and A is conflict-free, it holds that $\{b\} \models_R \overline{a}$. Thus by definition, $\{b\}$ reversely $<$-attacks A, and moreover item 2 holds.

□

Next we show that to decide $<$-stable assumption sets, one needs to only consider normal $<$-attacks and reverse $<$-attacks to singleton assumption sets. From this and Lemma 2 it

follows that one can define $<$-stable semantics with forward-derivability. We will make use of this fact in our encoding for $<$-stable semantics in Section 7.

Proposition 4. Let $D = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework. A conflict-free set $E \subseteq \mathcal{A}$ is $<$-stable if and only if for all $b \in \mathcal{A}$ that are not normally $<$-attacked by E, either $b \in E$ or $\{b\}$ is reversely $<$-attacked by E.

Proof. Let $E \subseteq \mathcal{A}$ be a conflict-free assumption set. First, assume that E is $<$-stable. Suppose for contradiction that there is an assumption $\{b\}$ that is not normally $<$-attacked by E such that $\{b\} \nsubseteq E$, and $\{b\}$ is not reversely $<$-attacked by E. Then $\{b\}$ is neither $<$-attacked by E nor a subset of E, contradicting E being $<$-stable.

Now assume that for all $\{b\}$ that are not normally $<$-attacked by E, either $\{b\} \subseteq E$ or $\{b\}$ is reversely $<$-attacked by E. It immediately follows by definition that E is $<$-stable. □

We will show that that the presence of $<$-attacks between two sets of assumptions can be decided in polynomial time in Proposition 5, using Lemma 3. Then Proposition 5 will be used to show an algorithm for finding the $<$-grounded assumption set in an ABA$^+$ framework that satisfies WCP, also giving an upper bound to the complexity of this problem (in Proposition 6).

Lemma 3 ([47])**.** Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework, $x \in \mathcal{L}$, and A, A' be two sets of assumptions such that $A' \subseteq A$ and $A \vdash_{\mathcal{R}} x$. Further, let $G = (V, E)$ be a directed graph with

- $V = Th_{\mathcal{R}}(A)$, and
- $E = \{(a, b) \mid \exists r \in \mathcal{R} : body(r) \subseteq Th_{\mathcal{R}}(A), a \in head(r), b \in body(r)\}$.

It holds that $X \models_R x$ with $X \subseteq A$ and $X \cap A' \neq \emptyset$ if and only if there is a directed path from x to a node in A' in G.

Proposition 5. Let $(\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework and $A, B \subseteq \mathcal{A}$. One can decide in polynomial time whether A

1. normally $<$-attacks B, or
2. reversely $<$-attacks B.

Proof. Item 1: Consider for each $b \in B$ the set of assumptions in A that are not less preferred than b, that is $A_b = \{a \in A \mid a \not< b\}$. It directly follows from the definition of $<$-attack that A_b normally $<$-attacks $\{b\}$ if and only if $A_b \vdash_{\mathcal{R}} \overline{b}$. Both constructing A_b for each b and checking if $A_b \vdash_{\mathcal{R}} \overline{b}$ for any A_b can be done in polynomial time. Checking if A attacks B can be done based on A_b in polynomial time: if A_b normally $<$-attacks $\{b\}$, then A normally $<$-attacks B, and if A normally $<$-attacks B, there is a $b \in B$ such that A_b normally $<$-attacks $\{b\}$.

Item 2: consider the deductive closure of B, that is $Th_{\mathcal{R}}(B)$. Check for each $a \in A$ such that $\overline{a} \in Th_{\mathcal{R}}(B)$ whether $B' \models_R \overline{a}$ for some $B' \subseteq B$ such that $\exists b' \in B$ with $b' < a$. Due to Lemma 3, this can be reduced to checking reachability in a directed graph, which can be computed in polynomial time. To see how Lemma 3 applies, notice that we are interested in checking if $B' \models \overline{a}$ with $B' \subseteq B$ and $B' \cap B_{<a} \neq \emptyset$, where $B_{<a}$ is the set of assumptions of B that are less preferred than a.

□

Finally, we show that in an ABA$^+$ framework that satisfies WCP, the $<$-grounded assumption set can be computed with an algorithm that makes polynomially many calls to an NP-oracle. The algorithm uses the def_F operator of ABA$^+$, similarly to how one can compute the grounded assumption set of an ABA framework with the def_F operator (recall Proposition 1). We describe an implementation of this algorithm in Section 7. In particular, we implement the def_F operator with iterative ASP calls and use this to find the $<$-grounded assumption set.

Proposition 6. In an ABA$^+$ framework that satisfies WCP, the $<$-grounded assumption set can be computed via a deterministic polynomial time algorithm that can access an NP-oracle.

Proof. Let $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$ be an ABA$^+$ framework that satisfies WCP. Due to [17, proof of Theorem 4.7 (iv)], it holds that the least fixed point of def_F is equal to the $<$-grounded assumption set. Consider the subproblem of checking whether a given set of assumptions X defends a singleton assumption set $\{a\}$. We claim that this problem is in coNP. To see this, consider the complementary problem, namely, checking whether X does not defend $\{a\}$. This can be solved by the following procedure. First non-deterministically construct an assumption set $Y \subseteq \mathcal{A}$. Check whether Y $<$-attacks $\{a\}$ (this is decidable in polynomial time due to Proposition 5). If Y does $<$-attack $\{a\}$, check whether X $<$-attacks Y (which is again decidable in polynomial time). If not, then by definition X does not defend $\{a\}$. In other cases X (Y does not $<$-attack $\{a\}$ or X $<$-attacks Y), X does defend $\{a\}$. There are polynomially many assumptions, and thus $def_F(X)$ can be computed with polynomial many calls (at most $|\mathcal{A}|$) to an NP-oracle following the preceding procedure. The $<$-grounded assumption set is found after a polynomial number of applications (at most $|\mathcal{A}|$) of def_F starting with $\emptyset$. Overall the NP-oracle is accessed no more than $|\mathcal{A}|^2$ times. □

5 Answer Set Programming

We briefly recall answer set programming (ASP) [42, 52, 10, 39], the approach to logic programming implementing the stable model semantics (also called answer set semantics).

ASP is a prominent declarative approach to solving NP-complete search and optimization problems. Declarative programming refers to an approach in which a problem is encoded in a specified language, the encoded instance is solved with a solver for that language, and finally the result is interpreted in terms of the original problem. In this case, we represent our ABA reasoning problems as ASP problems and use an ASP solver to solve them.

5.1 Answer Set Programs

An ASP program is a collection of rules of the form

$$h \leftarrow b_1, \ldots, \ b_k, \ \mathit{not} \ b_{k+1}, \ldots, \ \mathit{not} \ b_m.$$

where h and each b_i is an atom. An atom b_i is a predicate $p(t_1, \ldots, t_n)$ with each t_j either a constant or a variable. An answer set program, a rule, and an atom, respectively, is ground if it is free of variables. The process of making a non-ground program ground is called grounding or variable replacement.

A rule is positive if $k = m$ (no negated atoms), a fact if $m = 0$ (no body), and a constraint if there is no head h. A fact can be shortened to "h." by omitting the arrow. A constraint

$$\leftarrow b_1, \ldots, \ b_k, \ \mathit{not} \ b_{k+1}, \ldots, \ \mathit{not} \ b_m.$$

is a shorthand for

$$x \leftarrow b_1, \ldots, \ b_k, \ \mathit{not} \ b_{k+1}, \ldots, \ \mathit{not} \ b_m, \ \mathit{not} \ x.,$$

where x is a ground atom not occurring anywhere else. Intuitively, the head of a fact must hold and at least one body element of a constraint must not hold in a solution to the program.

The answer sets to a program are defined in terms of a grounded program. Let π be an answer set program that contains variables. Let GP be the set of rules obtained by applying all possible substitutions from the variables to the set of constants appearing in the program. GP is ground. An interpretation I consists of a subset of the ground atoms. I satisfies a positive rule $r = h \leftarrow b_1, \ldots, b_k$ if and only if the presence of all positive body elements $b_1, \ldots, b_k$ in I implies that the head atom is in I. That is, I satisfies r if and only if

$$\{b_1, \ldots, b_k\} \subseteq I \text{ implies } h \in I.$$

For a program π consisting only of positive rules, let $Cl(\pi)$ be the subset-minimal interpretation I that satisfies all rules in π. That is, $Cl(\pi) = I$ if and only if I, but not any

proper subset of I, satisfies all rules in π. The reduct of a program π with respect to an interpretation I is

$$\pi^I = \{(h \leftarrow b_1, \ldots, b_k) \mid (h \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m) \in \pi, \{b_{k+1}, \ldots, b_m\} \cap I = \emptyset\}.$$

In other words, the reduct contains the positive part of all rules of π, except the rules whose negative part contains the negation of an atom that is contained in I. An interpretation I is an answer set of a ground program π if $I = Cl(\pi^I)$ where π^I is the reduct. An interpretation I is an answer set of a non-ground program π if I is an answer set of GP of π. A program is satisfiable if the program has at least one answer set and unsatisfiable otherwise.

5.2 ASP Solvers and Additional Features

Variables are highly useful for modelling problems, but the answer sets are defined in terms of ground programs. Thus solving problems with ASP typically consists of modelling the problem with variables, grounding the program, and finally solving the ground program. We will now discuss the last two steps, as well as some additional language features we make use of later.

Grounders use various techniques to make the grounding more efficient than the naive method of simply creating all substitutions of variables by constants. These techniques involve partial evaluation, rewriting and database technology. The most popular grounders are DLV [49], Lparse [56], and gringo [40, 41]. DLV also incorporates a solver, and Lparse was originally paired with the solver Smodels.

Arguably the state-of-the-art ASP solver today is clasp [37]. Algorithmically clasp is similar to modern conflict-driven clause learning (CDCL) SAT solvers [55], notably implementing conflict-driven learning and backjumping. In addition clasp incorporates various optimizations used by modern SAT solvers. Unlike SAT solvers, clasp also ensures that the given output solutions are answer sets.

In this work, we will use the state-of-the-art ASP system clingo [40, 41], incorporating gringo as the grounder and clasp as the ASP solver. Using a system that incorporates both grounding and solving in the same system streamlines ASP solving. Clingo grounds and computes answer sets to a given program without the need for any further pre- or post-processing. One can specify how many answer sets are computed. In particular, clingo can compute one answer set and terminate (useful for credulous reasoning), or compute every answer set (useful for sceptical reasoning and σ-assumption set enumeration).

Clingo and systems built on top of it support a host of additional features, some of which are very useful for implementing reasoning in ABA. Firstly, we will use the cautious enumeration mode of clingo in our algorithm for finding the ideal assumption set. The cautious mode returns the intersection of all answer sets to a given program [40]. We also use asprin [11], a framework for optimization in ASP that relies on iterative calls to an ASP solver (by default clingo). Specifically we use the capabilities of asprin to get subset-maximal answer sets of a program with respect to a predicate of arity one. Formally, I is an optimal answer set if

there is no answer set J such that $\{p(t) \mid p(t) \in I\} \subset \{p(t) \mid p(t) \in J\}$). This is useful for capturing the preferred semantics via ASP.

6 ASP for Reasoning in ABA

In this section we present algorithms for reasoning in ABA based on ASP. In particular, we provide algorithms for deciding the credulous and sceptical acceptability of sentences as well as enumerate σ-assumption sets under the ABA semantics admissible, complete, preferred, stable, and ideal semantics. For all of these except ideal semantics, it holds that A is an assumption set of a given ABA framework F if and only if there is an answer set M of $\pi_\sigma \cup \texttt{ABA}(F)$ where $A = \{a \mid \mathbf{in}(a) \in M\}$, π_σ is the ASP encoding for semantics σ, and $\texttt{ABA}(F)$ is the encoding of F. For ideal semantics, we present a more complex algorithm making use of the ASP encodings.

Firstly we need an encoding of a given ABA framework, in particular the assumptions, rules and contraries of the framework. Assume an ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$ with $\mathcal{R} = \{r_1, \ldots, r_n\}$. The framework is represented in ASP as the facts

$$\begin{aligned}\texttt{ABA}(F) = &\{\mathbf{assumption}(a). \mid a \in \mathcal{A}\} \cup \\ &\{\mathbf{head}(i, b). \mid r_i \in \mathcal{R}, b \in \mathit{head}(r_i)\} \cup \\ &\{\mathbf{body}(i, b). \mid r_i \in \mathcal{R}, b \in \mathit{body}(r_i)\} \cup \\ &\{\mathbf{contrary}(a, b). \mid b = \overline{a}, a \in \mathcal{A}\}.\end{aligned}$$

The predicates for assumptions and contraries are unary and binary, respectively, as expected. The predicate $\mathbf{assumption}(a)$ is interpreted as a being an assumption. $\mathbf{contrary}(a, b)$ indicates that b is the contrary of a. The rules are expressed via separate predicates for heads and bodies of rules, with a unique rule index linking them. The predicate $\mathbf{head}(i, b)$ is interpreted as b being the head of the rule with index i. Similarly $\mathbf{body}(i, b)$ indicates that b is contained in the body of the rule with index i.

Example 9. Consider an ABA framework F with

$$\begin{aligned}\text{sentences } \mathcal{L} &= \{a, b, x, y\}, \\ \text{assumptions } \mathcal{A} &= \{a, b\}, \\ \text{contraries} \quad &\overline{a} = y,\ \overline{b} = x, \\ \text{rules } \mathcal{R} &= \{(x \leftarrow a, y), (y \leftarrow b)\}.\end{aligned}$$

F encoded as ASP is

$$\begin{aligned}\texttt{ABA}(F) = \ & \textbf{assumption}(a).\\ & \textbf{assumption}(b).\\ & \textbf{head}(1, x).\\ & \textbf{body}(1, a).\\ & \textbf{body}(1, y).\\ & \textbf{head}(2, y).\\ & \textbf{body}(2, b).\\ & \textbf{contrary}(a, y).\\ & \textbf{contrary}(b, x).\end{aligned}$$

△

Towards encoding the reasoning problems, consider the ASP module π_{common} (Listing 6.1), which is a part of most of our encodings. The module derives all conflict-free assumption sets of the given ABA framework. In the first two lines, the assumptions are partitioned into **in** and **out** via a non-deterministic guess, so that each possible partitioning is considered. We will refer to the assumptions that are assigned to **in** as A. The next three lines give the deductive closure of A: the predicate **supported** holds for all sentences that can be forward-derived from A. This is done in the following way. The third line encodes the base case, namely that an assumption is always said to derive itself. In the fourth line, whenever a rule is "triggered", meaning that all of the sentences in the rule body are supported by **in**, the head of the rules if also supported. The fifth line encodes the triggering of rules via the ASP conditional construct. The conditional $\textbf{supported}(X) : \textbf{body}(R, X)$ holds when for each $\textbf{body}(R, X)$ of the given rule index R, the ASP atoms $\textbf{supported}(X)$ are present. * Thus the rule is triggered when each sentence in its body is supported. The sixth rule then gives the assumptions that are attacked (or defeated) by A, making use of the deductive closure **supported** of A. The last line is a constraint to enforce conflict-freeness: an assumption can not be both **in** and defeated by **in**.

Given an ASP program, such as π_{common}, additional constraints can be placed to encode credulous and sceptical reasoning. One can specify a queried sentence by adding the rule

$$\textbf{query}(s).$$

for a sentence $s \in \mathcal{L}$ to the input framework. Now adding the rule

$$\leftarrow \mathit{not}\, \textbf{supported}(X), \textbf{query}(X).$$

to an ASP encoding (given that the encoding uses the predicate **supported** in the manner used in π_{common}) rules out answers where x is not derived from the σ-assumption set, giving

* Conditional literals allow for a succinct presentation of a rule with a variable sized body. See [38] for more information.

Listing 6.1: Module π_{common}

```
1 in(X) ← assumption(X), not out(X).
2 out(X) ← assumption(X), not in(X).
3 supported(X) ← assumption(X), in(X).
4 supported(X) ← head(R,X), triggered_by_in(R).
5 triggered_by_in(R) ← head(R,_), supported(X) : body(R,X).
6 defeated(X) ← supported(Y), contrary(X,Y).
7 ← in(X), defeated(X).
```

the answers to a credulous query. Similarly, to answer sceptical reasoning, one can look for counterexamples by adding the rule

$$\leftarrow \mathbf{supported}(X), \mathbf{query}(X).$$

Now if there are no answer sets when this constraint is included, there are no σ-answer sets where the query is not derivable from it, meaning that the query is sceptically accepted.

6.1 Stable, Admissible, Complete, and Preferred Semantics

The ASP encodings for stable, admissible, complete and preferred semantics are built on top of π_{common}. The simplest one is stable semantics. It suffices to add the rule

$$\leftarrow \mathbf{out}(X), not\, \mathbf{defeated}(X).$$

to π_{common}, to enforce that no assumption can be both **out** and not defeated by **in**. This satisfies the definition of stable semantics: all assumptions are either in the stable assumption set, or attacked by it.

For admissible semantics, recall Proposition 3, stating that a conflict free assumption set A is admissible if and only if the set of assumptions not attacked by A does not attack A. This is implemented in Listing 6.2 by checking whether a contrary of any assumption that is **in** is derivable from the assumptions not attacked by **in**. The first three lines of the encoding mirror the Lines 3 to 5 of π_{common}, only now the deductive closure of a different set is computed. Line 4 then gives the assumptions attacked by the assumptions undefeated by **in**, and the last line states that an assumption can not be **in** and attacked by the undefeated assumptions. Thus the admissible semantics is encoded by joining π_{common} with π_{adm}.

The complete semantics only needs an additional constraint on top of the admissible encoding. The rule

$$\leftarrow \mathbf{out}(X), not\, \mathbf{attacked\_by\_undefeated}(X).$$

enforces that no assumption can be out of the complete assumption set A and not attacked by the assumptions that A does not attack. In other words, if A is a complete assumption set, then if A defends an assumption, that assumption is in A. Complete semantics is encoded by this line added to π_{common} and π_{adm}.

For preferred semantics, we make use of asprin [11], which provides tools for optimization in ASP. Recall that an assumption set is preferred if it is a maximal admissible (or complete) assumption set. The optimization statements

$$\#\textbf{preference}(p1, \text{superset})\{\textbf{in}(X) : \textbf{assumption}(X)\}.$$
$$\#\textbf{optimize}(p1).$$

added to the encoding for admissible (or complete) semantics give preferred assumption sets. We use the encoding of admissible semantics for this in our experiments in Section 8. Concretely, the first line specifies the "preference" of having supersets of **in**, and the second line that this optimization shall be applied.

6.2 Ideal Semantics

The ideal assumption set can be computed via an iterative algorithm, shown in Algorithm 1. The algorithm is adapted from [26]. Recall that the ideal assumption set is the unique maximal admissible set that is a subset of every preferred assumption set. The algorithm computes an overapproximation of the ideal assumption set (Lines 1–4) and then iteratively removes assumptions not defended by the set (Lines 6–11), yielding the ideal assumption set as proved in [26, Theorem 8]. Concretely, on Lines 1 and 2 the set of assumptions contained in some admissible assumption set is computed, and on Lines 3 and 4 all assumptions attacked by this set are removed, resulting in A_{PSA}. This is an overapproximation of the ideal assumption set, as it has to be conflict-free and contain only assumptions that are in some admissible assumption set. The current overapproximation of the ideal assumption set is denoted as Γ. In the loop of Lines 6–11, Γ is refined into the ideal assumption set by repeatedly identifying (Line 9) the assumptions in Γ that are attacked by assumptions not attacked by Γ (computed on Line 8) and removing them from Γ (Line 10). Everything besides Line 1 can be computed in polynomial time and relatively easily. For Line 1, to obtain the set of assumptions not contained in any admissible assumption set, we make use of the ASP encoding for admissible semantics (see Section 6.1). This can be done by using the cautious reasoning mode of clingo, which gives the intersection of all answer sets of the program. The assumptions labelled **out** in the intersection constitute the needed set. Thus the algorithm can be implemented given the admissible encoding.

Remark 1. The algorithm as presented in [26] is erroneous on Lines 8 and 9. Instead of the Lines 8 and 9 of Algorithm 1, the original has (apart from slight variation in notation) the

Listing 6.2: Module π_{adm}

```
1 derived_from_undefeated(X) ← assumption(X), not defeated(X).
2 derived_from_undefeated(X) ← head(R,X), triggered_by_undefeated(R).
3 triggered_by_undefeated(R) ← head(R,_), derived_from_undefeated(X) : body(R,X).
4 attacked_by_undefeated(X) ← contrary(X,Y), derived_from_undefeated(Y).
5 ← in(X), attacked_by_undefeated(X).
```

lines

$$\Xi := \{a \in \mathcal{A}_{\text{out}} \mid \Gamma \not\vdash_{\mathcal{R}} \overline{a}\}$$
$$\Delta := \{a \in \Gamma_{\text{in}} \mid \Xi \cup \mathcal{A}_{\text{CA}} \vdash_{\mathcal{R}} \overline{a}\}.$$

In words, in the original Ξ only contains assumptions in $\mathcal{A}_{\text{out}}$ instead of all assumptions, and Δ contains assumptions attacked by $\Xi \cup \mathcal{A}_{\text{CA}}$ instead of ones attacked by Ξ. The effect is that the original algorithm only considers attacks from $\mathcal{A}_{\text{out}} \cup \mathcal{A}_{\text{CA}}$ when deciding if Γ defends itself. In other words, attacks from a set containing assumptions from $\mathcal{A}_{\text{in}} \setminus \mathcal{A}_{\text{CA}} = \mathcal{A}_{\text{PSA}}$ are not considered. This is not sufficient, since there can be attacks from $\mathcal{A}_{\text{out}} \cup \mathcal{A}_{\text{PSA}}$ to Γ. By definition $\mathcal{A}_{\text{PSA}}$ can not by itself attack Γ, but nothing prevents assumptions from $\mathcal{A}_{\text{PSA}}$ forming a strict subset of a set of assumptions attacking an assumption in Γ. Thus the original algorithm would fail to identify some assumptions in Γ that are not defended by Γ, namely ones that are attacked by a set containing assumptions from $\mathcal{A}_{\text{out}}$ as well as $\mathcal{A}_{\text{PSA}}$.

Consider the following example where the original algorithm fails and our formulation succeeds.

$$\begin{aligned} \text{sentences } \mathcal{L} &= \{a, b, c, d, \overline{a}, \overline{b}, \overline{c}, \overline{d}\} \\ \text{assumptions } \mathcal{A} &= \{a, b, c, d\} \\ \text{rules } \mathcal{R} &= \{(\overline{b} \leftarrow a), (\overline{a} \leftarrow b), (\overline{c} \leftarrow a), (\overline{c} \leftarrow b), (\overline{d} \leftarrow c, d)\} \end{aligned}$$

There are two preferred extensions, $\{a, d\}$ and $\{b, d\}$. To see this, simply note that from both a and b one can derive the contrary of all other assumptions except d. Thus either of these assumptions joined with d attack all assumptions outside it and thus are subset-maximally admissible. The ideal assumption set is the maximal admissible set that is a subset of every preferred assumption set. In our example, the intersection of the preferred assumption sets

Algorithm 1 Computing the ideal assumption set [26, Algorithm 3]

Require: ABA framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-)$
Ensure: return the *ideal*-assumption set of F

$\mathcal{A}_{\text{out}} := \{a \in \mathcal{A} \mid a$ not credulously accepted under admissible semantics$\}$
$\mathcal{A}_{\text{in}} := \mathcal{A} \setminus \mathcal{A}_{\text{out}}$
$\mathcal{A}_{\text{CA}} := \{a \in \mathcal{A} \mid \mathcal{A}_{\text{in}} \vdash_{\mathcal{R}} \overline{a}\}$
$\mathcal{A}_{\text{PSA}} := \mathcal{A}_{\text{in}} \setminus \mathcal{A}_{\text{CA}}$
$\Gamma := \mathcal{A}_{\text{PSA}}$
repeat
    $\Gamma_{\text{in}} := \Gamma$
    $\Xi := \{a \in \mathcal{A} \mid \Gamma \not\vdash_{\mathcal{R}} \overline{a}\}$
    $\Delta := \{a \in \Gamma_{\text{in}} \mid \Xi \vdash_{\mathcal{R}} \overline{a}\}$
    $\Gamma := \Gamma \setminus \Delta$
until $\Gamma_{\text{in}} = \Gamma$
return Γ

is $\{d\}$. This set is not admissible, since $\{c, d\}$ attacks it without it defending itself. Thus the ideal assumption set is empty.

However, the algorithm as presented originally would return a non-empty assumption set. In the first 5 lines we get $\mathcal{A}_{\text{out}} = \{c\}$, $\mathcal{A}_{\text{in}} = \{a, b, d\}$, $\mathcal{A}_{\text{CA}} = \{a, b\}$, $\mathcal{A}_{\text{PSA}} = \{d\}$, and $\Gamma = \{d\}$. Then in the loop, $\Gamma_{in} = \{d\}$. On Line 8 the original algorithm checks which assumptions in $\mathcal{A}_{\text{out}}$ are not attacked by Γ. We get $\Xi = \{c\}$, since c is the only member of $\mathcal{A}_{\text{out}}$, and it also is not attacked by Γ. On Line 9 any member of Γ that is attacked by $\Xi \cup \mathcal{A}_{\text{CA}}$ is collected to Δ. In our case $\Xi \cup \mathcal{A}_{\text{CA}} = \emptyset$, and since d is not attacked by the empty set, $\Delta = \emptyset$. Finally on Line 10, $\Gamma \setminus \emptyset = \Gamma$, so on Line 11 we terminate and on Line 12 return that $\{d\}$ is the ideal assumption set.

In contrast, Algorithm 1 finds the correct ideal assumption set. The steps are identical until Line 8, when in our formulation Ξ is set to contain every assumption that $\{d\}$ does not attack, which is every assumption in the framework. Since the contrary of d is derivable from Ξ, on Line 9, Δ is set to contain d. Thus on Line 10, d is removed from Γ and the loop is executed again, after which the empty set is found to be the ideal assumption set.

7 ASP for Reasoning in ABA$^+$

In this section we address the problem of reasoning in ABA$^+$ by using ASP. Recall that there is a complexity jump between ABA and ABA$^+$ for many semantics, and thus algorithms for ABA$^+$ problems are expected to be more complex. We introduce algorithms for $<$-stable and $<$-grounded semantics, the former due to there being no complexity jump from ABA stable to ABA$^+$ $<$-stable, and the latter due to Proposition 6 showing an algorithm that makes polynomially many calls to an NP-oracle (ASP effectively serves as an NP-oracle for us).

For using ASP to reason in ABA$^+$, we need to give the preference relation along with the rest of the framework. For ABA$^+$ framework $F = (\mathcal{L}, \mathcal{R}, \mathcal{A}, ^-, \leq)$, let

$$\texttt{ABA}^+(F) = \texttt{ABA}(F) \cup \{\textbf{preferred}(x, y). \mid y \leq x\}.$$

The transitivity of the preference relation must be encoded separately, as shown in the module π_{prefs} given in Listing 7.1.

7.1 $<$-Stable Semantics

We provide an ASP encoding for $<$-stable semantics, similar to the encoding for stable semantics of ABA. Recall Proposition 4 for a restatement of $<$-stable semantics: a conflict-free assumption set A is $<$-stable if and only if each assumption not normally $<$-attacked by A is either reversely $<$-attacked by A, or included in A. This definition of $<$-stable semantics can be encoded as ASP in a similar way to the ABA encodings. In particular, the encoding uses π_{common} to guess a set of assumptions and to check conflict-freeness. Additionally, each assumption that is normally $<$-attacked by the guessed assumption set is computed, and lastly for each assumption not normally $<$-attacked, it is checked whether they are reversely $<$-attacked. Then the answer set to this program corresponds to a $<$-stable assumption set. This method works whether the given framework supports WCP or not.

The encoding is given by conjoining π_{common} and π_{prefs} with π_{stb+} (Listing 7.2). In π_{stb+}, the first four lines derive all assumptions normally $<$-attacked by A. This is achieved by extending the previously used **supported** predicate to take preferences into account. Concretely, the **preferredly_supported** predicate has a parameter, Y, for the assumptions which A might attack. For each Y, only the assumptions of A that are not less preferred than Y are taken into account when considering whether A derives the contrary of Y. Thus this predicate derives the normal $<$-attacks from A on Line 4. Lines 5–9 check whether A reversely $<$-attacks the assumptions that A does not normally $<$-attack. This is done by checking if these assumptions by themselves attack assumptions in A so that the attacked assumption in A is more preferred. Lines 5–7 compute what the normally $<$-undefeated assumptions derive by themselves. Line 8 computes which assumptions in A each of these assumptions attack. Based on this and the preference relation, Line 9 computes which of these assumptions A reversely

Listing 7.1: Module π_{prefs}

```
1  preferred(X,Z) ← preferred(X,Y), preferred(Y,Z).
2  strict_less_preferred(X,Y) ← preferred(Y,X), not preferred(X,Y).
3  n_less_preferred(X,Y) ← assumption(X), assumption(Y),
       not strict_less_preferred(X,Y).
```

Listing 7.2: Module π_{stb+}

```
1  preferredly_supported(X,Y) ← n_less_preferred(X,Y), assumption(X), in(X).
2  preferredly_supported(X,Y) ← head(R,X), preferredly_triggered_by_in(R,Y).
3  preferredly_triggered_by_in(R,Y) ← head(R,_), assumption(Y),
       preferredly_supported(X,Y) : body(R,X).
4  normally_defeated(Y) ← preferredly_supported(X,Y), contrary(Y,X).
5  derived_from_undef_assumption(Z,Z) ← assumption(Z), not normally_defeated(Z).
6  derived_from_undef_assumption(Y,Z) ← head(R,Y),
       trig_by_undef_assumption(R,Z).
7  trig_by_undef_assumption(R,Z) ← head(R,_), assumption(Z),
       derived_from_undef_assumption(Y,Z) : body(R,Y).
8  in_att_by_normally_undef_assumption(X,Z) ← in(X), contrary(X,Y),
       derived_from_undef_assumption(Y,Z).
9  reversely_defeated(Z) ← strict_less_preferred(Z,X),
       in_att_by_normally_undef_assumption(X,Z).
10 ← out(Y), not normally_defeated(Y), not reversely_defeated(Y).
```

<-attacks. Finally, Line 10 enforces the conditions of <-stable semantics: an assumption can not be outside of A and not normally nor reversely <-attacked by A.

7.2 <-Grounded Semantics

In this subsection we detail an algorithm for <-grounded semantics. The idea of the algorithm is the same as for grounded semantics: iteratively compute the defence-operator def_F until a fixed point. However, due to the higher complexity of <-grounded semantics, the algorithm requires iterative calls to an ASP program instead of one call.

In [17, proof of Theorem 4.7 (iv)] it is proven that the <-grounded assumption set of an ABA$^+$ satisfying WCP is the least fixed point of the operator $def_F(A)$*. To compute the <-grounded assumption set, we implement the algorithm implied by Proposition 6, using an ASP encoding as a subroutine. The encoding determines whether a given assumption is <-defended by a given set of assumptions, and so calling it for each assumption in the framework and the set A implements $def_F(A)$. The <-grounded assumption set can be computed by repeatedly calling this implementation of $def_F(A)$ by starting with $A_1 = \emptyset$ and updating so that $A_i = def_F(A_{i-1})$ until a fixed point. Then the final A_i is the <-grounded assumption

*Recall that $def_F(A)$ gives the set of assumptions that A <-defends.

set. The subroutine is given by joining $\pi_{grd+subroutine}$ (Listing 7.3) with π_{prefs}.

In addition to the framework $\texttt{ABA}^+(F)$, the encoding takes as input the currently $<$-defended set of assumptions via the predicate **def** and the assumption whose status as being $<$-defended is being questioned via the predicate **target**. The encoding non-deterministically guesses a set of assumptions, called suspects, and checks whether they are not $<$-attacked by **def**, and do $<$-attack **target**. The techniques used in this encoding are similar to the previously introduced ones, especially π_{stb+}. A main difference is that in $\pi_{grd+subroutine}$ the suspect set is guessed instead of the potential σ-assumption set, and there are a few more conditions to check. On Lines 3–6 it is determined whether the guessed suspects normally $<$-attack the target. On Lines 7–11 it is checked whether the suspects reversely $<$-attack the target, by checking if from the target one can derive a contrary of a more preferred suspect. Next is checking if the defending set (**def**) defends the target. Lines 12–15 decide which of the suspects are normally $<$-attacked by the defending set. To check if the defending set reversely $<$-attacks the suspect set, the reachability procedure of Proposition 5 is implemented on Lines 16–22. Finally in the last three lines, the necessary constraints are placed. Lines 23 and 24 enforce that the suspect must not be normally or reversely $<$-attacked by the defending set. Line 25 enforces that the target must be either normally or reversely $<$-attacked by the suspect set. Thus the program is satisfiable if and only if there is some set of assumptions that $<$-attack the target without the defending set attacking it. In other words, the defending set $<$-defends the target if and only if the program is unsatisfiable.

Listing 7.3: Module $\pi_{grd+\ subroutine}$

```
1  susp(X) ← assumption(X), not other(X).
2  other(X) ← assumption(X), not susp(X).
3  preferredly_supported_by_susps(X) ← target(Y), n_less_preferred(X,Y),
       assumption(X), susp(X).
4  preferredly_supported_by_susps(X) ← head(R,X),
       preferredly_triggered_by_susps(R).
5  preferredly_triggered_by_susps(R) ← head(R,_),
       preferredly_supported_by_susps(X) : body(R,X).
6  target_normally_attacked ← target(Y), preferredly_supported_by_susps(X),
       contrary(Y,X).
7  derived_from_target(X) ← target(X).
8  derived_from_target(X) ← head(R,X), triggered_by_target(R).
9  triggered_by_target(R) ← head(R,_), derived_from_target(X) : body(R,X).
10 susp_attacked_by_target(X) ← susp(X), contrary(X,Y), derived_from_target(Y).
11 target_revers_attacked ← target(Y), strict_less_preferred(Y,X),
       suspect_attacked_by_target(X).
12 supported_by_def(X,Y) ← susp(Y), n_less_preferred(X,Y), assumption(X), def(X).
13 supported_by_def(X,Y) ← susp(Y), head(R,X), triggered_by_def(R,Y).
14 triggered_by_def(R,Y) ← head(R,_), assumption(Y),
       supported_by_def(X,Y) : body(R,X).
15 susp_normally_defeated_by_def ← supported_by_def(X,Y), contrary(Y,X).
16 supported_by_susps(X) ← assumption(X), susp(X).
17 supported_by_susps(X) ← head(R,X), triggered_by_susps(R).
18 triggered_by_susps(R) ← head(R,_), supported_by_susps(X) : body(R,X).
19 reachable(X,Y) ← triggered_by_susps(R), head(R,Y), body(R,X).
20 reachable(X,Y) ← reachable(X,Z), reachable(Z,Y).
21 reachable(X,X) ← susp(X).
22 susp_revers_defeated_by_def ← def(Y), contrary(Y,X),
        susp(Z), reachable(Z,X), strict_less_preferred(Z,Y).
23 ← susp_normally_defeated_by_def.
24 ← susp_revers_defeated_by_def.
25 ← not target_normally_attacked, not target_revers_attacked.
```

8 Experiments

We tested the performance of our ASP-based approach to reasoning in ABA and ABA$^+$ against the current state-of-the-art systems (recall Section 3) across a variety of semantics and reasoning tasks. Our experiments cover a comparison between our approach and previous systems, as well as a scalability experiment of our system separately. To obtain a comprehensive picture of the performance of the ASP approach compared to other systems, we ran tests for each semantics for which we introduced an ASP-based solution in this work: admissible, complete, stable, preferred, ideal for ABA, and $<$-grounded and $<$-stable for ABA$^+$.

We compare the ASP approach to aba2af and abagraph for every semantics that our system and either of those systems support: both of them support admissible semantics, and further aba2af supports stable and preferred semantics. For the rest of the semantics, namely complete and ideal as well as $<$-stable and $<$-grounded, we compare the ASP approach to ABAplus. We replicated the chosen reasoning tasks (credulous or sceptical reasoning or σ-assumption set enumeration) for a given semantics from the previous ABA studies [14, 46] when applicable. For the semantics not covered in those experiments, the choice of reasoning task was dictated by the capabilities of the different systems (recall Section 3).

For the comparison experiments, we replicated the main experiment of [46], the enumeration of all solutions with respect to a query under admissible semantics. Due to abagraph working by deriving a justification for a single sentence, we additionally tested credulous acceptance under admissible semantics to enable the best performance of abagraph. These two are the only problems that both aba2af and abagraph support. Following [46], we tested sceptical acceptance under stable semantics, comparing our ASP approach to aba2af. For the rest of the ABA semantics, namely complete, preferred and ideal[†], σ-assumption set enumeration was tested. These task choices were largely forced by the restrictions of the other systems (recall Section 3). In particular, for ideal and complete semantics, ABAplus only supports σ-assumption set enumeration. For preferred semantics , our own approach forced the task of σ-assumption set enumeration: the asprin version we used does not support querying.

For the comparisons against ABAplus, namely the experiments for complete, ideal, $<$-stable, and $<$-grounded semantics, we tested σ-assumption set enumeration. This is because ABAplus supports only $<$-σ-assumption set enumeration[‡] directly.

We tested the scalability of the ASP approach under every semantics mentioned except $<$-grounded (as we lack a sufficiently efficient way to enforce WCP for large enough instances). The tested reasoning problem was primarily credulous acceptance, but under preferred and ideal semantics we tested σ-assumption set enumeration. For preferred semantics this is because the version of asprin we used does not support querying sentences. For ideal semantics,

[†]For ideal, this amounts to finding the unique ideal assumption set.

[‡]For frameworks that satisfy WCP, like the ones we use here, there is guaranteed to be a unique $<$-grounded assumption set, analogously to ABA. Therefore, again, enumerating $<$-grounded assumption sets amounts to finding the unique one.

there is a unique ideal assumption set, so querying the same framework for different sentences would amount to finding the same assumption set repeatedly.

Our approach compares very favourably to the other systems, and in fact solves all of the tested problems very quickly in the benchmarks we used for the comparisons. The ASP approach scales to significantly bigger instances than the ones used for the comparison. The instances used for the comparison have less than 100 sentences, but our approach can routinely solve problems in frameworks with 1000-3000 sentences, depending on the semantics and reasoning task. Our approach did not time out on instances with less than 1000 sentences in any of our experiments.

As the ASP solver for both our approach and translation-based system aba2af we used clingo v5.2.2 [41]. Abagraph was run on SICStus Prolog version 4.5. The experiments were ran on 2.83-GHz Intel Xeon E5440 quad-core machines with 32-GB RAM. We enforced a 600-second timeout limit for each instance. In the rest of this section, we explain the various benchmarks used in our experiments, detail the comparison between the ASP approach and the other systems, and lastly review the scalability of our approach on larger instances.

8.1 Benchmarks

Our benchmarks consist of five different benchmark sets.

1. The 680 frameworks* introduced in [14], and also used in [46], with up to 90 sentences per framework.
2. 120 frameworks we generated with between 10 to 30 sentences per framework.
3. Same as Item 2, but augmented with a preference relation.
4. 100 frameworks we generated with between 50 and 4000 sentences per framework.
5. 60 frameworks we generated including a preference relation, with between 50 and 4000 sentences per framework.

Benchmarks 1–3 were used for the comparison between the ASP approach and the other systems. Specifically, benchmarks 1 were used for the experiments under admissible, stable, and preferred semantics. Benchmarks 2 were used for the comparisons under complete and ideal semantics, and benchmarks 3 for the ABA$^+$ comparisons under $<$-stable and $<$-grounded semantics. Benchmarks 4 and 5 were used for the scalability experiments: benchmarks 4 for the ABA problems and benchmarks 5 for the ABA$^+$ problems. We created benchmark sets 2 and 3 because benchmark set 1 does not conform to the additional restrictions that ABAplus has for its input frameworks (such as an assumption not being a contrary of another assumption). We offer the details on all benchmark sets in this subsection.

*http://robertcraven.org/proarg/experiments.html

The benchmark set 1 contains 680 ABA frameworks, a single framework containing up to 90 sentences. Due to the high number of trivial instances in the benchmark set, as noted in [46], we followed the setup of that study in removing trivial instances from consideration. Instances where the queried sentence is either derivable from the empty set, or not derivable at all, are trivial under certain semantics. Sentences derivable from empty body are always credulously accepted under admissible semantics and sceptically accepted under stable semantics. Non-derivable sentences on the other hand are not credulously accepted under admissible semantics. These conditions can be checked in polynomial time from the rules, and thus are not interesting when testing the performance of the systems on NP-hard problems.

The total number of instances for the comparison experiments is affected by the filtering of trivial instances and the choice between querying multiple sentences per framework or not. Recall that benchmark set 1 was used for the comparison experiments under admissible, stable, and preferred semantics. For the experiments on queried sentences, there are 1728 final instances for credulous reasoning under admissible, and 4613 for sceptical reasoning under stable semantics. For enumeration of preferred assumption sets, all 680 of the base frameworks were used as instances.

For benchmark sets 2–5, we largely followed the benchmark generating process of [14]. We picked the parameter family 4 from [14] (with some modifications) because in that family each parameter is dependent on the number of sentences, instead of being fixed like in the other families. This way the frameworks scale naturally, without having, for instance, the absolute number of assumptions fixed. The parameters we use are as follows. Following [14], we fixed 37% of the sentences to be assumptions. Compared to the original benchmarks, we made some modifications to the number of rules deriving each sentence and the sizes of rule bodies. In particular, we set each sentence to be derivable with at least one rule, and ensured that no sentence is derivable from the empty set, to avoid trivial instances. Moreover, we limit the number of rules deriving any one sentence to 20, and also limit the size of any rule body to 20. This is so that when the instances grow very large in terms of sentences, the ways and complexities of deriving a sentence does not blow up. The intuition is to prioritise having a large number of "arguments" over the number of sentences in them, as well as not having disproportionate redundancy in how many ways a single sentence can be derived. This results in the following parameters. Let n_s denote the number of sentences in a framework. The number of rules deriving each sentence is randomly chosen from the interval $[1, \min(n_s/7, 20)]$, and the size of each rule body is randomly chosen from the interval $[1, \min(n_s/8, 20)]$. Thus the number of sentences is a varying parameter from which the number of assumptions, the number of rules per sentence, and the size of rule bodies depend on.

For the experiments on ABA$^+$ $<$-stable and $<$-grounded semantics, we augmented the benchmarks with preference relations. To also test the potential effect of the density of the relation on solving time, we used two different preference relations. We chose a random permutation of the assumptions $(a_i)_{0<i\leq n}$, and for each $i < j$ assigned a_i to be preferred to a_j with a fixed probability of 15% or 40%. Another consideration is WCP (see Section 2.4), which ABAplus requires to hold for its input frameworks. If WCP does not hold in a given framework,

ABAplus modifies the framework to enforce WCP, which can also be time-consuming. To provide a fair comparison, we let ABAplus enforce WCP for each of the benchmark frameworks prior to running the actual experiments, and used these modified frameworks as test instances.

Using the parameters defined in the previous paragraphs, the benchmark set 2 contains a total of 120 frameworks with 10, 14, 18, 22, 26 and 30 sentences in them, 20 frameworks per number of sentences. Benchmark set 2 was used for the enumeration of complete and ideal semantics. Thus each framework constitutes one test instance, for a total of 120 instances.

The benchmark set 3 consists of the same frameworks augmented with preference relations so that half of the frameworks have a preference relation with density 15% and half a density of 40%. The set 3 was used for the enumeration of $<$-stable and $<$-grounded semantics, giving again a total of 120 instances.

The benchmark set 4 consists of 100 frameworks with the number of sentences being 50, 250, 500, and from there increasing by 500 until 4000, with ten frameworks per number of sentences. The benchmark set 4 was used for the scalability experiments on the ABA problems. Under admissible, complete and stable semantics, we queried the credulous acceptance of ten sentences per framework, giving 100 instances for each number of sentences, 1000 instances in total. Under preferred and ideal semantics, enumeration of σ-assumption sets was tested, giving 10 instances per number of sentences and 100 instances in total.

In the benchmark set 5, we have six frameworks per number of sentences, with again half of the instances having a preference density of 15% and one a density of 40%. These frameworks in general do not satisfy WCP. We tested credulous reasoning under $<$-stable semantics, querying ten sentences per framework. This gives 60 instances for each number of sentences and 600 instances in total.

8.2 Comparison with Existing Systems

In this section we detail the results of the comparison experiments between the ASP approach and the state-of-the-art systems. Recall that under admissible semantics, we tested credulous reasoning and the enumeration of solutions given a queried sentence. We tested sceptical reasoning under stable semantics. For preferred, complete, ideal, $<$-stable, and $<$-grounded, we tested σ-assumption set enumeration.

The results of the comparison between our ASP approach, aba2af, abagraph and ABAplus is shown in Table 8.1. The number of timeouts and the mean, median and cumulative running time over solved instances are shown. Note that different problems have a different number of instances, as seen in the caption of the table. The results for the best performing system for each problem is shown in bold. Across all semantics and tasks, the ASP approach clearly outperforms the other systems. There are no timeouts for ASP, and the cumulative times are a fraction of what the other systems exhibit. This is true for both ABA and ABA$^+$ experiments. To highlight the overall performance of ASP, consider the closest gap between it and any other system: in enumeration of preferred assumption sets, ASP has three fourths

Table 8.1: Runtime comparison. Mean (**mean**), median (**med.**) and cumulative running times (**cum.**) over solved instances, **#to** is the number of timeouts. Number of instances: 1728 (*adm*), 4613 (*stb*), 680 (*prf*), 120 (*com*, *ideal*, and ABA$^+$ <-*stb* and <-*grd*). The system performing best for each problem is shown in bold.

Problem	**Approach**	**#to**	**Running times** (s)		
			mean	**med.**	**cum.**
ABA *adm*,	**ASP**	**0**	**0.030**	**0.012**	**53**
enum. w/query	abagraph	401	15.170	1.064	20131
	aba2af	393	18.650	0.588	24897
ABA *adm*,	**ASP**	**0**	**0.018**	**0.012**	**31**
cred. accep.	abagraph	200	8.464	1.056	12932
	aba2af	364	13.990	0.572	19078
ABA *stb*,	**ASP**	**0**	**0.009**	**0.008**	**59**
scept. accep.	aba2af	648	10.942	1.040	43386
ABA *prf*	**ASP**	**0**	**0.333**	**0.328**	**226**
enum. wo/query	aba2af	255	6.082	0.464	2585
ABA *com*	**ASP**	**0**	**0.005**	**0.004**	**1**
enum wo/query	ABAplus	9	15.287	0.268	1697
ABA *ideal*	**ASP**	**0**	**0.025**	**0.024**	**3**
	ABAplus	18	22.490	0.322	2293
ABA$^+$ <-*stb*	**ASP**	**0**	**0.018**	**0.008**	**2**
enum. wo/query	ABAplus	9	15.583	0.268	1729
ABA$^+$ <-*grd*	**ASP**	**0**	**0.255**	**0.126**	**31**
	ABAplus	9	15.611	0.268	1732

of the median running time of aba2af, and aba2af indeed times out on more than a third of the instances, while ASP solves all instances. The cumulative running time is also ten times higher for aba2af than for ASP.

8.3 Scalability of the ASP Approach

As can be seen from Table 8.1, the benchmarks used in the comparison experiments are quite trivial for ASP. To further study the scalabaility of the ASP approach, we separately tested ASP on larger instances, described in Section 8.1. The overall results are shown in Table 8.2 in terms of timeouts and mean running times over solved instances, and in Figure 8.1 in terms of median running times over all instances. The reasoning problem under preferred and ideal semantics is σ-assumption set enumeration. Under the other semantics, credulous acceptance was tested.

As expected, the ASP approach can solve significantly larger instances than used in the comparison experiments. For some semantics and tasks, even some of the instances with 4000 sentences are solved. In general, instances with between 2000 and 3000 sentences are being routinely solved for ABA, with some variation depending on the semantics and task. To confirm the superior performance of the ASP approach, we ran abagraph on these

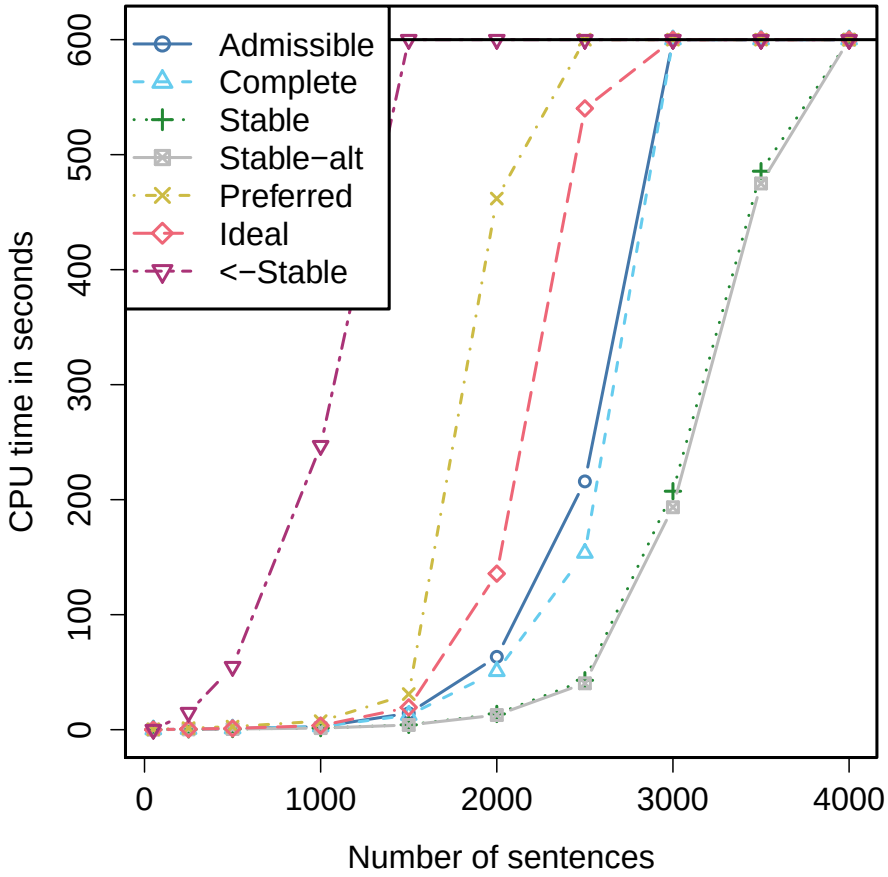

Figure 8.1: Median running times per number of sentences in the scalability experiments for each semantics. Timeouts are included as a running time of 600 seconds.

scalability instances for credulous acceptability under admissible semantics. It could solve the 50 sentence instances, but none of the larger ones.

We also tested the encoding introduced in [12] for stable semantics. This encoding of stable semantics is referred to as *stb*-alt in Table 8.2 and Stable-alt in Figure 8.1. The very similar results between our stable encoding and the stable encoding introduced in [12], seen in both Table 8.2 and Figure 8.1, suggests the two encodings are effectively equivalent in terms of computational performance.

The ABA$^+$ problem of credulous acceptability under <-stable semantics seems empirically harder than the ABA problems, with a strict cutoff in the ability of ASP to solve instances between 1000 and 1500 sentences, lower than for any other tested problem. This is a significant difference between ABA and ABA$^+$, especially since the complexity of this problem is the same (namely NP-complete) as for credulous reasoning under admissible, complete and stable semantics. In our experiments, the density of preferences had no apparent impact on the performance of our system: the instances with a preference relation density of 15% and 40% had nearly identical running times and the same number of timeouts.

Table 8.2: Scalability of ASP on larger frameworks.
Number of instances per $|\mathcal{L}|$: 100 (*adm*, *com*, *stb*, *stb*-alt), 10 (*prf*, *ideal*), 60 (<-*stb*).

	#timeouts (mean running time over solved (s))													
$\lvert\mathcal{L}\rvert$	**ABA** *adm*		**ABA** *com*		**ABA** *stb*		**ABA** *stb*-alt		**ABA** *prf*		**ABA** *ideal*		**ABA**$^+$ <-*stb*	
50	0	(0.01)	0	(0.01)	0	(0.01)	0	(0.01)	0	(0.3)	0	(0.03)	0	(0.1)
250	0	(0.4)	0	(0.4)	0	(0.3)	0	(0.3)	0	(1.3)	0	(0.5)	0	(12.6)
500	0	(0.8)	0	(0.9)	0	(0.6)	0	(0.6)	0	(2.8)	0	(1.2)	0	(53.1)
1000	0	(2.9)	0	(2.9)	0	(1.4)	0	(1.4)	0	(8.7)	0	(4.1)	0	(241.0)
1500	0	(13.8)	0	(12.3)	0	(4.3)	0	(4.1)	0	(32.4)	0	(18.2)	60	(0.0)
2000	0	(99.2)	0	(75.3)	0	(19.1)	0	(18.3)	5	(144.5)	0	(155.8)	60	(0.0)
2500	22	(126.1)	10	(201.1)	0	(74.2)	0	(72.7)	7	(268.8)	4	(292.5)	60	(0.0)
3000	70	(173.6)	58	(222.9)	18	(173.6)	18	(163.8)	10	(0.0)	9	(463.6)	60	(0.0)
3500	85	(211.3)	79	(253.6)	48	(231.8)	47	(228.1)	10	(0.0)	10	(0.0)	60	(0.0)
4000	89	(108.1)	87	(135.2)	81	(158.1)	81	(154.4)	10	(0.0)	10	(0.0)	60	(0.0)

8.4 Discussion

It seems clear that using ASP to reason directly over ABA and ABA$^+$ frameworks offers a significant performance advantage over current specialized algorithms for solving ABA and ABA$^+$ reasoning problems. For the systems that translate ABA frameworks to AFs, the major bottleneck may be the translation itself, instead of the AF reasoning part done via ASP. This was observed in the case of aba2af in [46]. In those experiments, the AF translation took over 80% of the overall runtime for most of the instances. The results of Table 8.1 suggest a similar phenomenon for ABAplus. Notice how ABAplus has nearly identical performance under complete, $<$-stable and $<$-grounded semantics. This suggests the possibility of a shared bottleneck, most likely in the translation phase. In fact the presence of preferences (when WCP is already satisfied) does not significantly contribute to the baseline running time. These observations motivate further focus on direct declarative approaches to reasoning in ABA, as well as in other structured argumentation formalisms. In abstract argumentation, SAT solvers along with ASP solvers are among the best-performing approaches to practical reasoning in the formalism [13].

More evaluation of the effects of different properties of ABA frameworks on the computational hardness of reasoning in them is needed. For example, in our experiments here, and in the experiments in [14, 46], the frameworks are quite similar to each other. It is not known how manipulating different parameters of ABA frameworks affect the hardness of reasoning in them. For example, in most frameworks used in this study, the number of assumptions is 37% of sentences, and it is not clear how changing this ratio would affect the runtime of the different systems. For ABA$^+$, another question is how WCP affects the empirical performance of ABA reasoning systems. A potential future experiment would use large instances where half of the frameworks satisfy WCP, and compare the time required to answer reasoning problems between the two classes of frameworks.

9 Conclusions

Assumption-based argumentation (ABA) is a central structured argumentation formalism with numerous applications. In this thesis, we studied the computational aspects of reasoning in ABA. The main results of the work are new, significantly faster practical algorithms for solving key reasoning problems in ABA and ABA$^+$. The algorithms are based on encodings to answer set programming (ASP) and the use of efficient ASP solvers. In particular, our approach is able to solve the problems of credulous and sceptical reasoning as well as σ-assumption set enumeration under admissible, complete, stable, preferred, and ideal semantics, as well as the corresponding problems for ABA$^+$ under $<$-stable and $<$-grounded semantics. Most of these problems are NP-hard. We showed empirically that our approach outperforms the state-of-the-art reasoning systems. Our work is, to our knowledge, the first to implement ABA reasoning directly with declarative tools. In addition, we provided new computational complexity results of reasoning in ABA$^+$.

The development of efficient algorithms for reasoning in structured argumentation is important in realizing the full potential of computational argumentation, as structured formalisms offer representational advantages over abstract argumentation. This work is a significant advancement in this direction, and also encourages further study towards efficient algorithms for the remaining, presumably computationally harder ABA problems (especially for ABA$^+$), as well as to similar tools for different structured argumentation formalisms. In particular it seems likely that implementing the rest of the ABA$^+$ semantics in a declarative manner will improve on the performance of current state of the art. Since many ABA$^+$ are on a higher complexity level than NP, iterative calls to an ASP solver can be used to solve them. An ASP solver is already used as an NP-oracle in our algorithm for $<$-grounded semantics, and developing similar algorithms for the other semantics seems possible. For abstract argumentation, such algorithms are shown to be efficient: in [27], a system for solving AF reasoning problems on polynomial hierarchy levels beyond NP (including sceptical reasoning under preferred semantics) with an iterative SAT-based algorithm is detailed, improving on the efficiency of previous state-of-the-art systems. Due to the theoretical similarity between many structured argumentation formalisms — for example ABA has been shown to be translatable to a fragment of ASPIC$^+$ [54] — there is hope that other structured argumentation formalisms beside ABA can be encoded as ASP in a similar manner. This may improve on the performance of state-of-the-art systems for the other formalisms as well.

Acknowledgements

I would like to thank my supervisors Associate Professor Matti Järvisalo and Dr. Johannes P. Wallner for all their excellent help during and before the writing of this thesis. I also thank Andreas Niskanen for his helpful comments. This work was done as part of the Constrained Reasoning and Optimization group of the Department of Computer Science at University of Helsinki. I am grateful for the financial support from the Academy of Finland (grants 312662 and 322869) and Helsinki Institute for Information Technology.

Bibliography

[1] K. Atkinson, P. Baroni, M. Giacomin, A. Hunter, H. Prakken, C. Reed, G. R. Simari, M. Thimm, and S. Villata. "Towards Artificial Argumentation". In: *AI Magazine* 38.3 (2017), pp. 25–36.

[2] Z. Bao, K. Cyras, and F. Toni. "ABAplus: Attack Reversal in Abstract and Structured Argumentation with Preferences". In: *Proc. PRIMA*. Ed. by B. An, A. L. C. Bazzan, J. Leite, S. Villata, and L. W. N. van der Torre. Vol. 10621. 2017, pp. 420–437.

[3] P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre, eds. *Handbook of Formal Argumentation*. College Publications, 2018.

[4] T. Bench-Capon and P. Dunne. "Argumentation in artificial intelligence". In: *Artif. Intell.* 171 (Jan. 2007), pp. 619–641.

[5] P. Besnard and A. Hunter. "A Review of Argumentation Based on Deductive Arguments". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 9, pp. 437–484.

[6] P. Besnard and A. Hunter. *Elements of Argumentation*. MIT Press, 2008. ISBN: 978-0-262-02643-7.

[7] P. Besnard, A. J. García, A. Hunter, S. Modgil, H. Prakken, G. R. Simari, and F. Toni. "Introduction to structured argumentation". In: *Argument & Computation* 5.1 (2014), pp. 1–4.

[8] A. Biere, M. Heule, H. van Maaren, and T. Walsh. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[9] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. "An Abstract, Argumentation-Theoretic Approach to Default Reasoning". In: *Artificial Intelligence* 93 (1997), pp. 63–101.

[10] G. Brewka, T. Eiter, and M. Truszczynski. "Answer set programming at a glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103.

[11] G. Brewka, J. P. Delgrande, J. Romero, and T. Schaub. "asprin: Customizing Answer Set Preferences without a Headache". In: *Proc. AAAI*. Ed. by B. Bonet and S. Koenig. AAAI Press, 2015, pp. 1467–1474.

[12] M. Caminada and C. Schulz. "On the Equivalence between Assumption-Based Argumentation and Logic Programming". In: *Journal of Artificial Intelligence Research* 60 (2017), pp. 779–825.

[13] F. Cerutti, S. A. Gaggl, M. Thimm, and J. P. Wallner. "Foundations of Implementations for Formal Argumentation". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 15, pp. 688–767.

[14] R. Craven and F. Toni. "Argument graphs and assumption-based argumentation". In: *Artificial Intelligence* 233 (2016), pp. 1–59.

[15] R. Craven, F. Toni, and M. Williams. "Graph-Based Dispute Derivations in Assumption-Based Argumentation". In: *TAFA 2013 Revised Selected Papers*. Ed. by E. Black, S. Modgil, and N. Oren. Vol. 8306. 2013, pp. 46–62.

[16] R. Craven, F. Toni, C. Cadar, A. Hadad, and M. Williams. "Efficient Argumentation for Medical Decision-Making". In: *Proc. KR*. Ed. by G. Brewka, T. Eiter, and S. A. McIlraith. AAAI Press, 2012, pp. 598–602.

[17] K. Cyras. "ABA+: assumption-based argumentation with preferences". PhD thesis. Imperial College London, UK, 2017.

[18] K. Cyras and T. Oliveira. "Resolving Conflicts in Clinical Guidelines using Argumentation". In: *Proc. AAMAS*. Ed. by E. Elkind, M. Veloso, N. Agmon, and M. E. Taylor. IFAAMAS, 2019, pp. 1731–1739.

[19] K. Cyras and F. Toni. "Properties of ABA+ for Non-Monotonic Reasoning". In: *Proc. NMR*. Ed. by G. Kern-Isberner and R. Wassermann. 2016, pp. 25–34.

[20] K. Cyras, X. Fan, C. Schulz, and F. Toni. "Assumption-Based Argumentation: Disputes, Explanations, Preferences". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 7, pp. 365–408.

[21] Y. Dimopoulos, B. Nebel, and F. Toni. "On the computational complexity of assumption-based argumentation for default reasoning". In: *Artificial Intelligence* 141.1/2 (2002), pp. 57–78.

[22] P. M. Dung. "On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games". In: *Artificial Intelligence* 77.2 (1995), pp. 321–358.

[23] P. M. Dung, R. A. Kowalski, and F. Toni. "Assumption-based argumentation". In: *Argumentation in Artificial Intelligence*. Ed. by I. Rahwan and G. R. Simari. 2009, pp. 25–44.

[24] P. M. Dung, R. A. Kowalski, and F. Toni. "Dialectic proof procedures for assumption-based, admissible argumentation". In: *Artificial Intelligence* 170.2 (2006), pp. 114–159.

[25] P. M. Dung, F. Toni, and P. Mancarella. "Some design guidelines for practical argumentation systems". In: *Proc. COMMA*. Ed. by P. Baroni, F. Cerutti, M. Giacomin, and G. R. Simari. 2010, pp. 183–194.

[26] P. E. Dunne. "The computational complexity of ideal semantics". In: *Artificial Intelligence* 173.18 (2009), pp. 1559–1591.

[27] W. Dvořák, M. Järvisalo, J. P. Wallner, and S. Woltran. "Complexity-sensitive decision procedures for abstract argumentation". In: *Artificial Intelligence* 206 (2014), pp. 53 –78.

[28] W. Dvořák and P. E. Dunne. "Computational Problems in Formal Argumentation and their Complexity". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 14.

[29] U. Egly, S. A. Gaggl, and S. Woltran. "Answer-set programming encodings for argumentation frameworks". In: *Argument & Computation* 1.2 (2010), pp. 147–177.

[30] X. Fan and F. Toni. "On the Interplay Between Games, Argumentation and Dialogues". In: *Proc. AAMAS*. 2016, pp. 260–268.

[31] X. Fan, F. Toni, A. Mocanu, and M. Williams. "Dialogical two-agent decision making with assumption-based argumentation". In: *Proc. AAMAS*. Ed. by A. L. C. Bazzan, M. N. Huhns, A. Lomuscio, and P. Scerri. IFAAMAS/ACM, 2014, pp. 533–540.

[32] J. K. Fichte, M. Hecher, and A. Meier. "Counting Complexity for Reasoning in Abstract Argumentation". In: *Proc. AAAI*. 2019, pp. 2827–2834.

[33] D. Gaertner and F. Toni. "CaSAPI: a system for credulous and sceptical argumentation". In: *Proc. NMR*. Ed. by G. R. Simari and P. Torroni. 2007, pp. 80–95.

[34] S. A. Gaggl, T. Linsbichler, M. Maratea, and S. Woltran. "Design and results of the Second International Competition on Computational Models of Argumentation". In: *Artificial Intelligence* 279 (2020), p. 103193.

[35] A. J. García and G. R. Simari. "Argumentation Based on Logic Programming". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 8, pp. 409–435.

[36] A. J. García and G. R. Simari. "Defeasible Logic Programming: An Argumentative Approach". In: *Theory and Practice of Logic Programming* 4.1-2 (2004), pp. 95–138.

[37] M. Gebser, B. Kaufmann, and T. Schaub. "Conflict-driven answer set solving: From theory to practice". In: *Artificial Intelligence* 187-188 (2012), pp. 52 –89.

[38] M. Gebser, A. Harrison, R. Kaminski, V. Lifschitz, and T. Schaub. "Abstract gringo". In: *Theory and Practice of Logic Programming* 15.4-5 (2015), pp. 449–463.

[39] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[40] M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. T. Schneider. "Potassco: The Potsdam Answer Set Solving Collection". In: *AI Communications* 24.2 (2011), pp. 107–124.

[41] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and P. Wanko. "Theory Solving Made Easy with Clingo 5". In: *Technical Communications of ICLP*. OASICS. 2016, 2:1–2:15.

[42] M. Gelfond and V. Lifschitz. "The Stable Model Semantics for Logic Programming". In: *Proc. ICLP/SLP*. MIT Press, 1988, pp. 1070–1080.

[43] T. F. Gordon, H. Prakken, and D. Walton. "The Carneades model of argument and burden of proof". In: *Artificial Intelligence* 171.10 (2007). Argumentation in Artificial Intelligence, pp. 875 –896.

[44] *International Competition on Computational Models of Argumentation*. http://argumentationcompetition.org/. Accessed: 2019-11-27.

[45] A. Karamlou, K. Cyras, and F. Toni. "Complexity Results and Algorithms for Bipolar Argumentation". In: *Proc. AAMAS*. 2019, pp. 1713–1721.

[46] T. Lehtonen, J. P. Wallner, and M. Järvisalo. "From Structured to Abstract Argumentation: Assumption-Based Acceptance via AF Reasoning". In: *Proc. ECSQARU*. Ed. by A. Antonucci, L. Cholvy, and O. Papini. Vol. 10369. 2017, pp. 57–68.

[47] T. Lehtonen, J. P. Wallner, and M. Järvisalo. *Reasoning over Assumption-Based Argumentation Frameworks via Answer Set Programming*. Submitted manuscript.

[48] T. Lehtonen, J. P. Wallner, and M. Järvisalo. "Reasoning over Assumption-Based Argumentation Frameworks via Direct Answer Set Programming Encodings". In: *Proc. AAAI*. 2019, pp. 2938–2945.

[49] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. "The DLV system for knowledge representation and reasoning". In: *ACM Trans. Comput. Log.* 7.3 (2006), pp. 499–562.

[50] S. Modgil and H. Prakken. "A general account of argumentation with preferences". In: *Artificial Intelligence* 195 (2013), pp. 361–397.

[51] S. Modgil and H. Prakken. "Abstract Rule-Based Argumentation". In: *Handbook of Formal Argumentation*. Ed. by P. Baroni, D. Gabbay, M. Giacomin, and L. van der Torre. College Publications, 2018. Chap. 6, pp. 287–364.

[52] I. Niemelä. "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm". In: *Annals of Mathematics and Artificial Intelligence* 25.3-4 (1999), pp. 241–273.

[53] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[54] H. Prakken. "An abstract framework for argumentation with structured arguments". In: *Argument & Computation* 1.2 (2010), pp. 93–124.

[55] J. P. M. Silva, I. Lynce, and S. Malik. "Conflict-Driven Clause Learning SAT Solvers". In: *Handbook of Satisfiability*. 2009, pp. 131–153.

[56] P. Simons, I. Niemelä, and T. Soininen. "Extending and implementing the stable model semantics". In: *Artificial Intelligence* 138.1 (2002), pp. 181 –234.

[57] M. Thimm. "The Tweety Library Collection for Logical Aspects of Artificial Intelligence and Knowledge Representation". In: *Künstliche Intelligenz* 31.1 (2017), pp. 93–97.

[58] M. Thimm. "Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation". In: *Proc. KR*. Vienna, Austria, 2014.

[59] M. Thimm and S. Villata. "The first international competition on computational models of argumentation: Results and analysis". In: *Artificial Intelligence* 252 (2017), pp. 267 –294.

[60] F. Toni. "A generalised framework for dispute derivations in assumption-based argumentation". In: *Artificial Intelligence* 195 (2013), pp. 1–43.

[61] F. Toni. "A tutorial on assumption-based argumentation". In: *Argument & Computation* 5.1 (2014), pp. 89–117.

[62] T. Wakaki. "Assumption-Based Argumentation Equipped with Preferences and its Application to Decision Making, Practical Reasoning, and Epistemic Reasoning". In: *Computational Intelligence* 33.4 (2017), pp. 706–736.